

高等院校信息安全专业系列教材

教育部高等学校信息安全专业教学指导委员会
中国计算机学会教育专业委员会
共同指导

顾问委员会主任：沈昌祥　编委会主任：肖国镇

网络安全协议综合实验教程

杨浩淼 李洪伟 冉鹏 编著

http://www.tup.com.cn

Information Security

根据教育部高等学校信息安全专业教学指导委员会编制的《高等学校信息安全专业指导性专业规范》组织编写

清华大学出版社

普通高等教育“十一五”国家级规划教材
高等院校信息安全专业系列教材

网络安全协议综合实验教程

杨浩淼　李洪伟　冉　鹏　编著

清华大学出版社
北　京

内 容 简 介

本书作为信息安全方面的实验教材，介绍了信息安全方面一些最基本的实验内容。这些实验包括第 1 章 VMware 虚拟网络的构建；第 2 章 IPSec 基础实验；第 3 章 SSL 基础实验；第 4 章缓冲区溢出攻击初级实验；第 5 章 Radius 综合实验；第 6 章 IPSec 综合实验；第 7 章 OpenSSL 综合实验；第 8 章 VPN 综合实验；第 9 章基于身份加密算法的综合实验以及第 10 章信息探测综合实验。在利用本书做实验的时候，不需要购买防火墙、入侵检测、VPN 等硬件设备。

本书适合作为高等学校信息安全及相关专业本科学生作为信息安全相关课程的实验教材，也适合作为企事业单位、公司员工进行信息安全方面的教育培训和技术研讨用书。

图书在版编目(CIP)数据

网络安全协议综合实验教程/杨浩淼，李洪伟，冉鹏编著. —北京：清华大学出版社，2016
高等院校信息安全专业系列教材
ISBN 978-7-302-42496-3

Ⅰ. ①网…　Ⅱ. ①杨…　②李…　③冉…　Ⅲ. ①计算机网络－安全技术－通信协议－高等学校－教材　Ⅳ. ①TP393.08

中国版本图书馆 CIP 数据核字(2015)第 316466 号

责任编辑：张　民　李　晔
封面设计：常雪影
责任校对：梁　毅
责任印制：沈　露
出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954
印 刷 者：北京富博印刷有限公司
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：7.75　　**字　　数**：192 千字
版　　次：2016 年 6 月第 1 版　　**印　　次**：2016 年 6 月第 1 次印刷
印　　数：1～2000
定　　价：19.50 元

产品编号：037413-01

高等院校信息安全专业系列教材

编审委员会

出版说明

21 世纪是信息时代，信息已成为社会发展的重要战略资源，社会的信息化已成为当今世界发展的潮流和核心，而信息安全在信息社会中将扮演极为重要的角色，它会直接关系到国家安全、企业经营和人们的日常生活。随着信息安全产业的快速发展，全球对信息安全人才的需求量不断增加，但我国目前信息安全人才极度匮乏，远远不能满足金融、商业、公安、军事和政府等部门的需求。要解决供需矛盾，必须加快信息安全人才的培养，以满足社会对信息安全人才的需求。为此，教育部继 2001 年批准在武汉大学开设信息安全本科专业之后，又批准了多所高等院校设立信息安全本科专业，而且许多高校和科研院所已设立了信息安全方向的具有硕士和博士学位授予权的学科点。

信息安全是计算机、通信、物理、数学等领域的交叉学科，对于这一新兴学科的培养模式和课程设置，各高校普遍缺乏经验，因此中国计算机学会教育专业委员会和清华大学出版社联合主办了“信息安全专业教育教学研讨会”等一系列研讨活动，并成立了“高等院校信息安全专业系列教材”编审委员会，由我国信息安全领域著名专家肖国镇教授担任编委会主任，指导“高等院校信息安全专业系列教材”的编写工作。编委会本着研究先行的指导原则，认真研讨国内外高等院校信息安全专业的教学体系和课程设置，进行了大量前瞻性的研究工作，而且这种研究工作将随着我国信息安全专业的发展不断深入。经过编委会全体委员及相关专家的推荐和审定，确定了本丛书首批教材的作者，这些作者绝大多数都是既在本专业领域有深厚的学术造诣，又在教学第一线有丰富的教学经验的学者、专家。

本系列教材是我国第一套专门针对信息安全专业的教材，其特点是：

① 体系完整、结构合理、内容先进。

② 适应面广。能够满足信息安全、计算机、通信工程等相关专业对信息安全领域课程的教材要求。

③ 立体配套。除主教材外，还配有多媒体电子教案、习题与实验指导等。

④ 版本更新及时，紧跟科学技术的新发展。

为了保证出版质量，我们坚持宁缺毋滥的原则，成熟一本，出版一本，并保持不断更新，力求将我国信息安全领域教育、科研的最新成果和成熟经验反映到教材中来。在全力做好本版教材，满足学生用书的基础上，还经由专

家的推荐和审定，遴选了一批国外信息安全领域优秀的教材加入本系列教材中，以进一步满足大家对外版书的需求。热切期望广大教师和科研工作者加入我们的队伍，同时也欢迎广大读者对本系列教材提出宝贵意见，以便对本系列教材的组织、编写与出版工作不断改进，为我国信息安全专业的教材建设与人才培养做出更大的贡献。

“高等院校信息安全专业系列教材”已于2006年年初正式列入普通高等教育“十一五”国家级教材规划(见教高[2006]9号文件《教育部关于印发普通高等教育“十一五”国家级教材规划选题的通知》)。我们会严把出版环节，保证规划教材的编校和印刷质量，按时完成出版任务。

2007年6月，教育部高等学校信息安全类专业教学指导委员会成立大会暨第一次会议在北京胜利召开。本次会议由教育部高等学校信息安全类专业教学指导委员会主任单位北京工业大学和北京电子科技学院主办，清华大学出版社协办。教育部高等学校信息安全类专业教学指导委员会的成立对我国信息安全专业的发展起到重要的指导和推动作用。2006年教育部给武汉大学下达了“信息安全专业指导性专业规范研制”的教学科研项目。2007年起该项目由教育部高等学校信息安全类专业教学指导委员会组织实施。在高教司和教指委的指导下，项目组团结一致，努力工作，克服困难，历时5年，制定出我国第一个信息安全专业指导性专业规范，于2012年年底通过经教育部高等教育司理工科教育处授权组织的专家组评审，并且已经得到武汉大学等许多高校的实际使用。2013年，新一届“教育部高等学校信息安全专业教学指导委员会”成立。经组织审查和研究决定，2014年以“教育部高等学校信息安全专业教学指导委员会”的名义正式发布《高等学校信息安全专业指导性专业规范》(由清华大学出版社正式出版)。“高等院校信息安全专业系列教材”在教育部高等学校信息安全专业教学指导委员会的指导下，根据《高等学校信息安全专业指导性专业规范》组织编写和修订，进一步体现科学性、系统性和新颖性，及时反映教学改革和课程建设的新成果，并随着我国信息安全学科的发展不断完善。

我们的E-mail地址：zhangm@tup.tsinghua.edu.cn；联系人：张民。

“高等院校信息安全专业系列教材”编审委员会

前 言

信息安全学科可分为狭义安全与广义安全两个层次，狭义的安全是建立在以密码论为基础的计算机安全领域，早期中国信息安全专业通常以此为基准，辅以计算机技术、通信网络技术与编程等方面的内容；广义的信息安全是一门综合性学科，从传统的计算机安全到信息安全，不但是名称的变更也是对内容的延伸，安全不再是单纯的技术问题，而是将管理、技术、法律等问题相结合的产物。

随着信息技术和计算机网络的普及，网络和信息安全对社会的生产，生活的影响越来越大。信息安全人才的培养和相关教材的编写也日益重要。本书从实践的角度出发，结合具体的实验来讲述信息安全的相关理论和技术。实验是对相关理论的运用和升华，作者希望借此给读者一个认识信息安全技术的新视角。本书的具体内容如下：

第 1 章介绍 VMware 虚拟网络的构建，这是进行实验的基础工作。一方面，进行 VMware 虚拟机的安装和虚拟网络的配置，这为后面的网络安全协议实验准备了虚拟网络实验环境；另一方面，对网络数据包的抓取软件 WireShark 进行了介绍，这是一个非常强大的网络协议分析工具。第 2 章介绍 IPSec 基础实验，通过本实验，将对 IPSec 的原理和协议运行机制有一定的了解；并熟悉 IPSec 数据包格式和协议流程。第 3 章介绍 SSL 的基础实验。第 4 章介绍缓冲区溢出攻击。在所有的漏洞中，缓冲区溢出漏洞占了远程网络攻击中的绝大多数。目前的缓冲区攻击方式有很多种，其中最常见的有栈溢出、堆溢出、整型溢出、格式化字符串溢出及文件流溢出等。本实验通过使用 Ollydbg 调试 OD 漏洞，理解缓冲区溢出攻击原理，了解栈溢出的攻击过程，从而在实际的写程序中尽量避免缓冲区溢出漏洞。第 5 章 Radius 基础实验的核心目的是，学会如何在 Windows Server 2003 的操作系统平台上进行安全配置以及服务器的搭建。而如何配置才能安全有效地实现 Radius 的远程客户端接入，是本实验最终的目的。第 6 章的 IPSec 综合实验是对 IPSec 相关内容的引申和提高。第 7 章主要进行 OpenSSL 综合实验，主要包括：编译 OpenSSL 源码包、使用 OpenSSL 创建 CA 实验以及 OpenSSL 的编程实验。第 8 章探讨如何开展 VPN 实验，这对提高学生计算机网络安全实践操作能力具有现实的指导意义。第 9 章介绍基于身份加密

算法的综合实验，本章在 MIRACL 大整数库的基础上，进行密码算法的实验。本章选用了比较新颖的基于身份公钥加密算法来进行实验。第 10 章介绍在攻击技术中，首要且不可或缺的一步是探测(Probe)。探测的主要目的便是对主机信息或者网络信息进行收集。本实验通过主机信息探测和安全漏洞探测两个方面来介绍两种著名的扫描工具，并以此了解其中的基本原理。

作　者

目 录

第 1 章 VMware 虚拟网络的构建

当前,虚拟机技术在计算机的各个领域得到了广泛应用。例如,在安全领域,可以利用虚拟机构建"蜜罐 (HoneyPot)"系统,对互联网上的网络攻击行为进行分析研究;在存储领域,可以利用虚拟机来减少服务器的数量,简化服务器的管理,将多种应用整合到单台服务器上完成;在机房建设领域,可以利用虚拟机技术实现机房机器的多种用途,而无须担心主机系统与硬件的损坏。

虚拟机对真正的计算机而言是一个概念,是一个模拟真实计算机进行工作的软件系统。虚拟机拥有自己的 CPU、指令系统、存储器组织、寄存器组、堆栈、输入输出等;可以接受指令系统的指令或用汇编语言及高级语言编写的程序,完成计算或数据处理工作。利用虚拟机技术生成的软件在功能上与使用的便利性上都与实际一致。

虚拟机在实验教学中的应用也越来越受到重视,本章基于 VMware 虚拟机在单机上构建了一个计算机网络实验平台。一方面,进行 VMware 虚拟机的安装和虚拟网络的配置,并为后面的网络安全协议实验准备了虚拟网络实验环境;另一方面,对网络数据包的抓取软件 WireShark 进行了介绍,这是一个非常强大的网络协议分析工具。

1.1 VMware 虚拟网络

1.1.1 虚拟机的简介

1. 虚拟机技术总述

虚拟机的种类很多,其中一个分支专门仿真特定的硬件,有模拟一个芯片时序逻辑的、模拟 CPU 指令的、模拟整个硬件开发板的以及模拟一个 PDA 的模拟器等;另一个分支可以仿真一个真实的 x86 计算机,能够在一台真实计算机上虚拟出另外一台计算机,同时运行两个或更多的操作系统。这类虚拟机软件通常需要一个被称作"主操作系统"(host OS)的操作系统作为底层基本平台,虚拟的操作系统就运行在主系统之上,通常称为"客户操作系统"(guest OS)。由于虚拟得到的是一个完全真实的计算机,所以主、客户系统可以实现互访,或者通过网络方式互访。客户系统可以访问主系统的网络系统,甚至能够实现 Internet 连接共享。如 VMWare、VirtualPC、twoOStwo、simics、VGS、Virtual Server、Cherry OS、SkyEye 等都属此类。

2. 常用虚拟机软件比较

1）VMWare

VMware 能够支持很多客户操作系统，能够模拟硬件包括主板、内存、硬盘（IDE 和 SCSI）、DVD/CD-ROM、软驱、网卡、声卡、串口、并口和 USB 口。但 VM Ware 没有模拟显卡。VMWare 为每一种 GuestOS 提供一个叫做 VMware-tools 的软件包，来增强 Guest OS 的显示和鼠标，以及同步虚拟机和 Host 的时间等功能。

VMware 提供了多种网络设置方式，例如 Bridged 方式、NAT 方式和 Host-only 方式。其中桥接方式下 Guest OS 的 IP 可设置为与 Host OS 在同一网段，GuestOS 相当于网络内的一台独立的机器，网络内的其他机器可访问 Guest OS，Guest OS 也可访问网络内的其他机器，当然与 Host OS 的双向访问也不成问题；在 NAT 方式下，也可以实现 Host OS 与 Guest OS 的双向访问。但网络内其他机器不能访问 Guest OS，Guest OS 可通过 Host OS 用 NAT 协议访问外界。

本书的虚拟网络环境是基于 VMware 的，因此对 VMware 虚拟网络，本书将在后面进一步介绍。

2）Virtual PC

Virtual PC 最早由 Connectix 公司推出，后被微软公司收购。Virtual PC 与 VMWare 的不同之处是：VMWare 要通过 VMware-tools 才能实现高分辨率和真彩色，而 Virtual PC 模拟了 S3 Trio 32/64（4MB 显存）这款比较通用的显卡，通用性很强；Virtual PC 的网络共享方式与 VMWare 不同，VMWare 是通过模拟网卡实现网络共享的，而 Virtual PC 是通过在现有网卡上绑定 Virtual PC emulated switch 服务实现网络共享的。

3）SkyEye

SkyEye 是一个指令级模拟器，可在通用的 Linux 和 Windows 平台上实现一个纯软件集成开发环境，模拟常见的嵌入式计算机系统；可运行多种嵌入式操作系统和各种系统软件，并可对它们进行源代码级的分析和测试，可模拟多种嵌入式开发板，支持多种 CPU 指令集。

不过，SkyEye 的目标不是验证硬件逻辑，而是协助开发、调试和学习系统软件，所以 SkyEye 与真实的硬件环境相比有一定差别。SkyEye 在时钟节拍的时序上不保证与硬件完全相同，对软件透明的一些硬件仿真进行了一定的简化，这样使 SkyEye 的执行效率更高。当然，SkyEye 并不能取代硬件的功能。

3. VMWare Workstation 软件包

VMware Workstation 是一款功能强大的桌面虚拟计算机软件，使用户可在单一的桌面上同时运行不同的操作系统，是进行开发、测试、部署新的应用程序的最佳解决方案。VMware Workstation 可在一部实体机器上模拟完整的网络环境，其更好的灵活性与先进的技术胜过了市面上大多数的虚拟计算机软件。对于企业的 IT 开发人员和系统管理员而言，VMware 在虚拟网路、实时快照、拖曳共享文件夹、支持 PXE 等方面的特点使它成为必不可少的工具。

本书的实验是在 VMware Workstation 7 上完成的，下面简单列举 VMware Workstation 7 新增的一些功能：

- 完善了对 3D 的支持。
- 支持最新 Windows 7 WDDM 驱动。
- 支持 vSphere 4.0 和 ESX。
- 可直接使用虚拟机进行打印。
- AutoProtect。
- 支持对虚拟机进行加密。
- 支持 IPv6、ALSA。
- 虚拟磁盘可扩展，无须使用额外的软件。

1.1.2　VMware 虚拟机的安装

1. VMWare Workstation 7.0 安装

VMWare Workstation 7.0 与大多数安装包一样，只需要一直单击安装向导界面的“下一步”按钮就可以完成 VMware 的安装。安装完成以后直接双击运行%VMware 安装目录%\vmware.exe。进入 VMware 界面以后选择“文件”→“新建”→“虚拟机”命令，正确操作完成以后会弹出一个名为“新建虚拟机向导”的窗口。

在新弹出的窗口上单击“下一步”按钮；虚拟机配置选择默认的“典型”，单击“下一步”按钮；客户机操作系统选择 Microsoft Windows，版本选择 Windows Server 2003 Enterprise Edition，单击“下一步”按钮；虚拟机名称和位置可以根据个人爱好自定义，单击“下一步”按钮；网络连接选择“使用 Host-only 网络”，单击“下一步”按钮；磁盘容量使用默认值，单击“完成”按钮。

至此，一台虚拟的个人计算机就已经准备就绪了。单击“编辑虚拟机设置”选项，可以手动配置虚拟机的硬件资源。值得注意的是，在此处配置的硬件资源需要与宿主计算机共享，所以分配的硬件资源应该根据实际情况而定。

2. 在虚拟机下安装 Windows 2003

选择在上一步安装好的虚拟机，单击“启动该虚拟机”选项；在 VMware 主窗口上选择“虚拟机”→“可移动设备”→CD-ROM →“编辑”按钮。此时会弹出一个名为“CD-ROM 设备”的子窗体。如果你准备的 Windows 2003 已经刻录在 CD 上，请将 CD 盘放入宿主计算机光驱，在子窗体选择“使用物理驱动器”选项，单击“确定”按钮。如果你准备的 Windows 2003 为 ISO 镜像文件，则在子窗体选择“使用 ISO 镜像”选项，单击“确定”按钮。如果此时虚拟机提示“…Operating System not found…”，在确保已经正确指向准备好的 Windows 2003 镜像的情况下，单击“关闭电源”按钮然后再次重启虚拟机。此时就可以看到熟悉的 Windows 系统安装向导。Windows 2003 安装结束以后，再次启动虚拟机，此时就可以看到熟悉的 Windows 启动画面。

至此虚拟机上安装 Windows 2003 已经结束。此时你可以用 VMware 提供的虚拟机克隆功能将刚刚安装好的 Windows 2003 进行克隆。

1.1.3 VMware 网络模式

VMware 提供了三种网络模式，它们是 bridged（桥接模式）、NAT（网络地址转换模式）和 host-only（主机模式）。

1. bridged

在这种模式下，VMware 虚拟出来的操作系统就像是局域网中的一台独立的主机，它可以访问网内任何一台机器。在桥接模式下，你需要手工为虚拟系统配置 IP 地址、子网掩码，而且还要和宿主机器处于同一网段，这样虚拟系统才能和宿主机器进行通信。同时，由于这个虚拟系统是局域网中的一个独立的主机系统，那么就可以手工配置它的 TCP/IP 配置信息，以实现通过局域网的网关或路由器访问互联网。使用桥接模式的虚拟系统和宿主机器的关系，就像连接在同一个 Hub 上的两台计算机。想让它们相互通信，就需要为虚拟系统配置 IP 地址和子网掩码，否则就无法通信。如果要利用 VMWare 在局域网内新建一个虚拟服务器，为局域网用户提供网络服务，就应该选择桥接模式。

2. host-only

在某些特殊的网络调试环境中，要求将真实环境和虚拟环境隔离开，这时就可采用 host-only 模式。在 host-only 模式中，所有的虚拟系统都是可以相互通信的，但虚拟系统和真实的网络是被隔离开的。

提示：在 host-only 模式下，虚拟系统和宿主机器系统是可以相互通信的，相当于这两台机器通过双绞线互连。在 host-only 模式下，虚拟系统的 TCP/IP 配置信息（如 IP 地址、网关地址、DNS 服务器等），都是由 VMnet1（host-only）虚拟网络的 DHCP 服务器来动态分配的。如果想利用 VMWare 创建一个与网内其他机器相隔离的虚拟系统，进行某些特殊的网络调试工作，就可以选择 host-only 模式。

3. NAT

使用 NAT 模式，就是让虚拟系统借助 NAT（网络地址转换）功能，通过宿主机器所在的网络来访问公网。也就是说，使用 NAT 模式可以实现在虚拟系统里访问互联网。NAT 模式下的虚拟系统的 TCP/IP 配置信息是由 VMnet8（NAT）虚拟网络的 DHCP 服务器提供的，无法进行手工修改，因此虚拟系统也就无法和本局域网中的其他真实主机进行通信。采用 NAT 模式最大的优势是虚拟系统接入互联网非常简单，不需要进行任何其他的配置，只需要宿主机器能访问互联网即可。如果想利用 VMWare 安装一个新的虚拟系统，在虚拟系统中不用进行任何手工配置就能直接访问互联网，那么建议采用 NAT 模式。

1.2 WireShark 网络数据包的抓取

【实验目的】

（1）学习使用 WireShark 软件的基本操作；

（2）了解 IP 数据报的格式，详细了解各部分的位宽和功能；

（3）抓取局域网络中的数据包并观察分析。

【实验内容】

使用 WireShark 进行网络抓包并观察 IP 包。

【实验设备与环境】

PC（Windows XP），WireShark 安装软件。

【实验方法步骤】

（1）观察 IP 包数据报格式，着重了解关键字段如位偏移等字段的长度和含义等。

（2）利用 WireShark 软件进行端口检测，抓 IP 包进行报文分析，如图 1-1 所示。

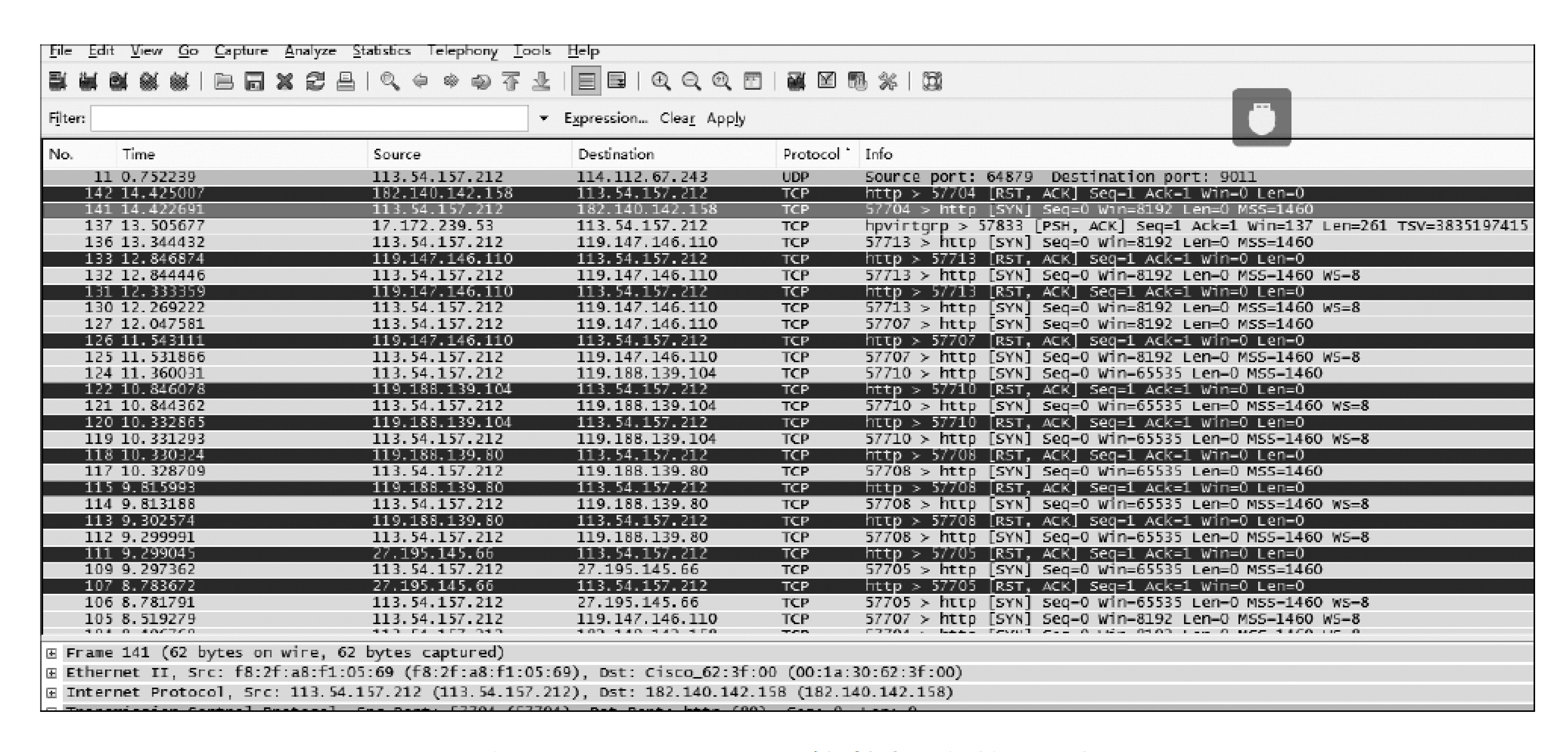


图 1-1　WireShark 软件抓取的 IP 包

分析该 IP 包可知：

起始时，源 IP 为 113.54.157.212 目的 IP 为 182.140.142.158 seq＝0 传送窗口为 8192 长度为 0；

返回时，源 IP 为 182.140.142.158 目的 IP 为 113.54.157.212 seq＝1 Ack＝1。

1.3　建立 VMWare 的虚拟网络环境的实验

【实验目的】

（1）掌握虚拟机 VMWare 的典型安装和配置方法；

（2）掌握基于虚拟机 VMware 的 Windows Server 2003 安装方法；

（3）熟悉并安装虚拟机 VMWare Tools 的主要功能。

【实验内容】

（1）安装 VMware Workstation 7；

（2）在虚拟机 VMware 上安装 Windows Server 2003。

【实验设备与环境】

（1）PC（Windows XP）；

（2）VMware Workstation7 安装软件；

（3）Windows Server 2003 系统镜像文件。

【实验方法步骤】

（1）安装 VMware Workstation 7。

（2）新建虚拟机并安装 Windows Server 2003 系统，如图 1-2 至图 1-8 所示。



图 1-2　选择典型配置

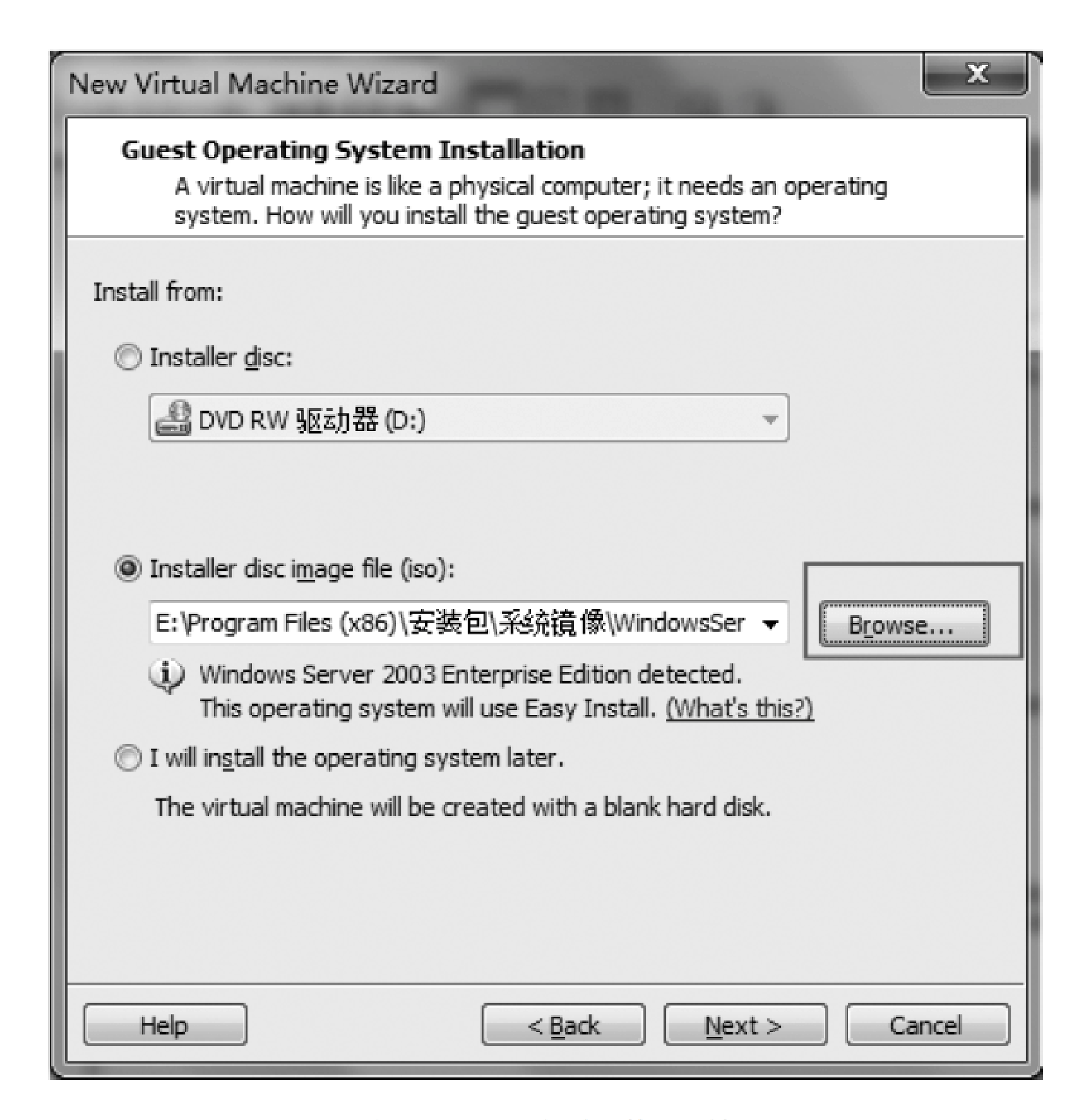


图 1-3　选择操作系统

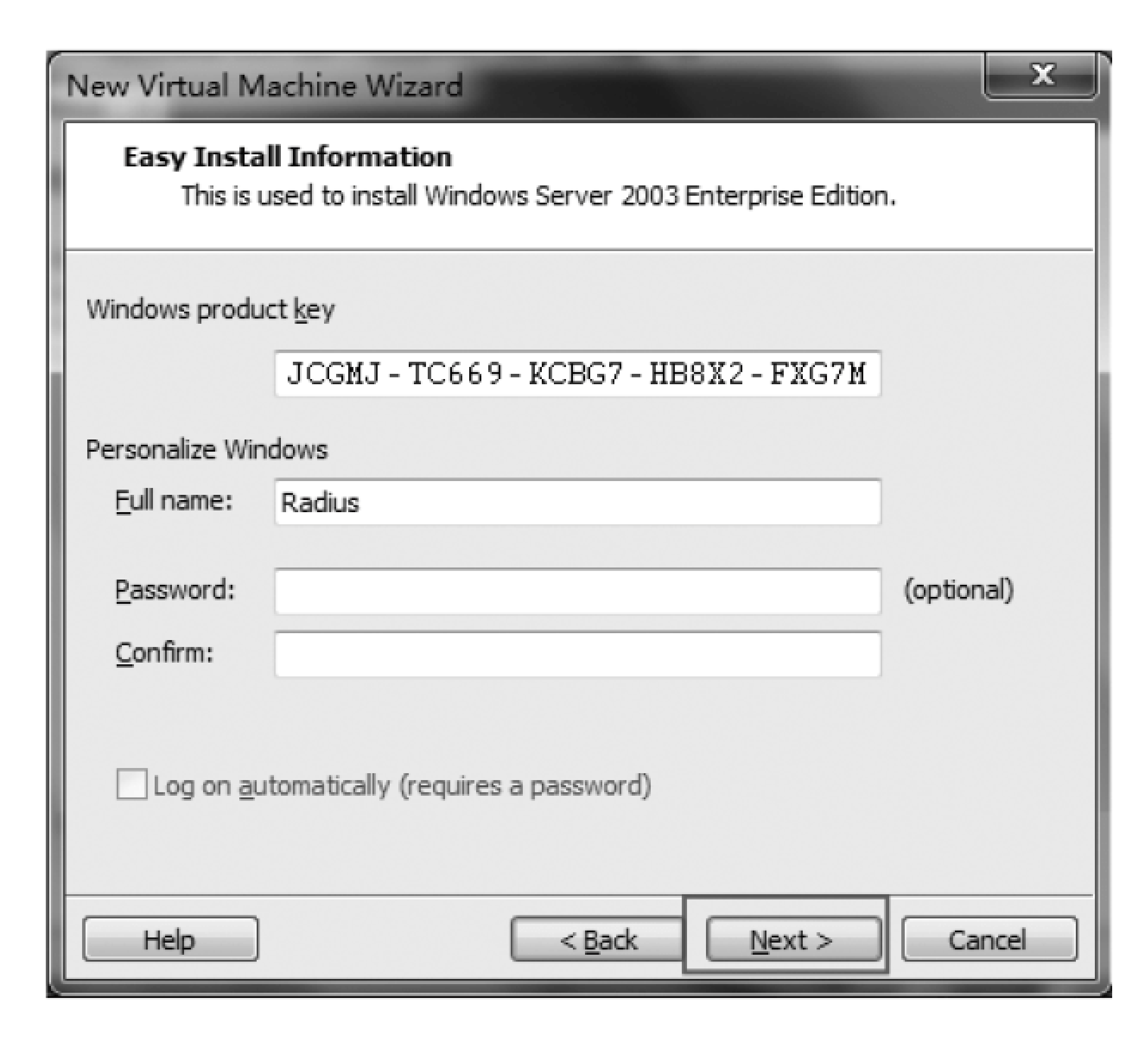


图 1-4　输入产品密钥



图 1-5　编辑虚拟机的位置

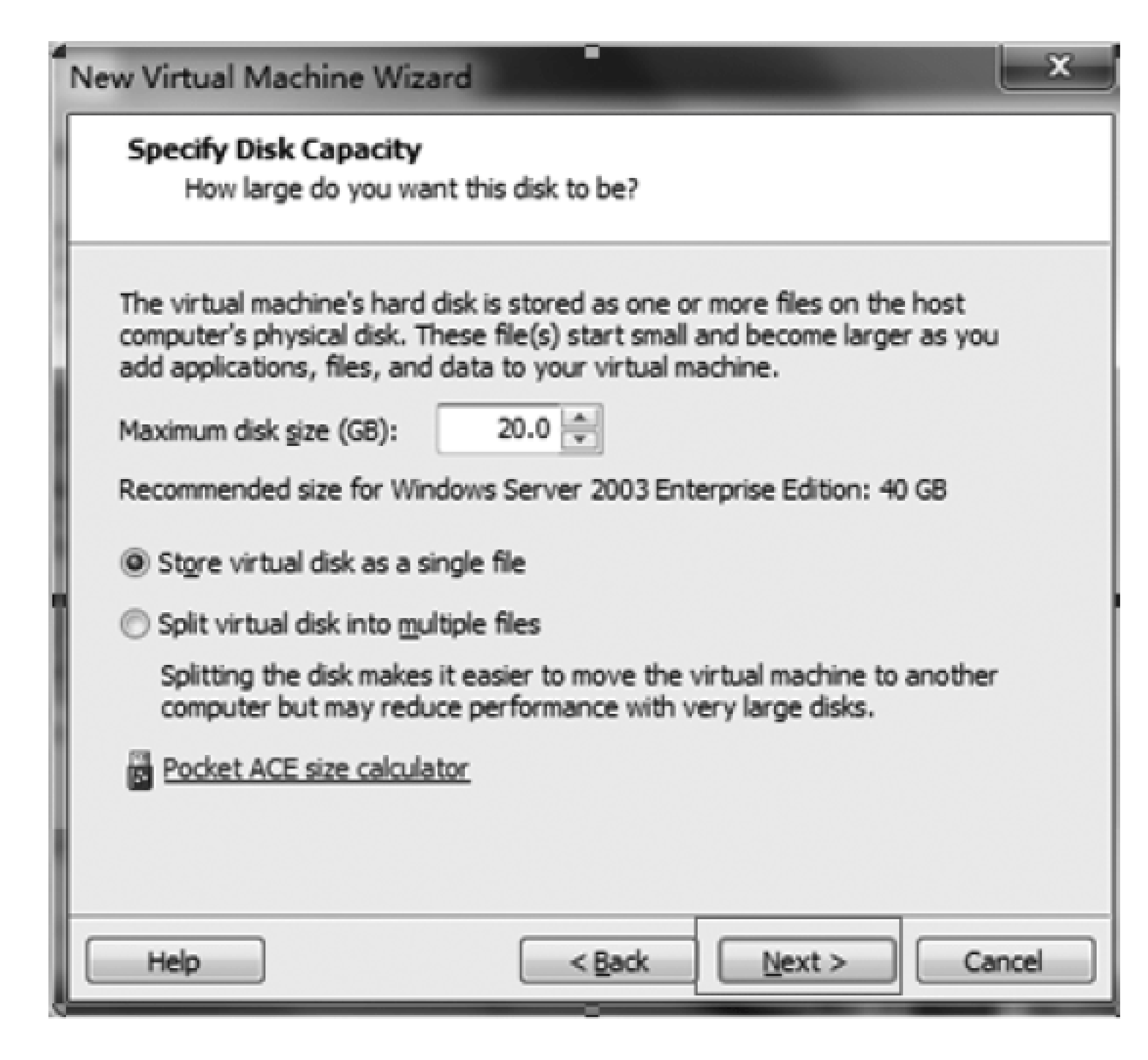


图 1-6　设置最大磁盘大小

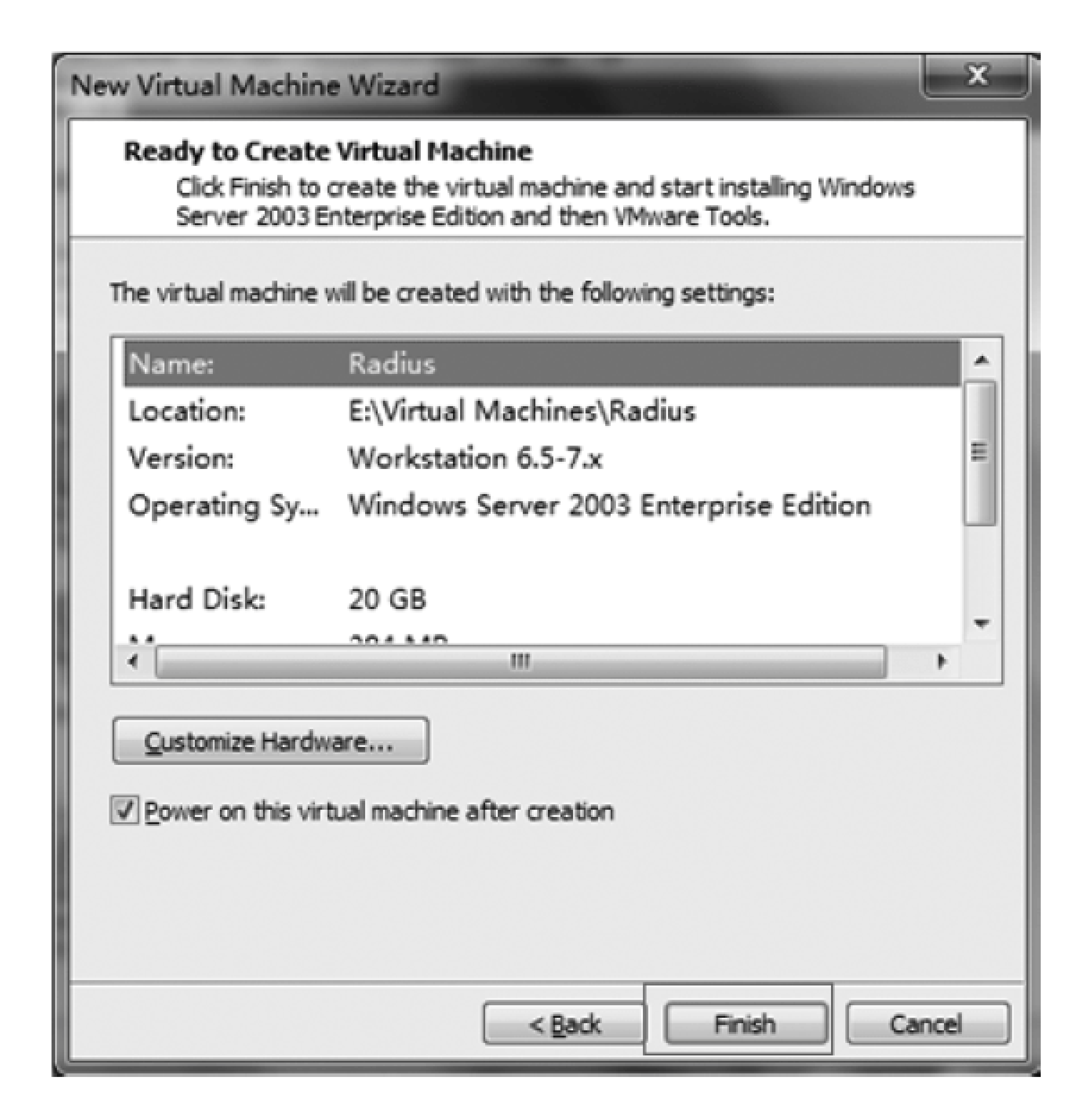


图 1-7　完成配置

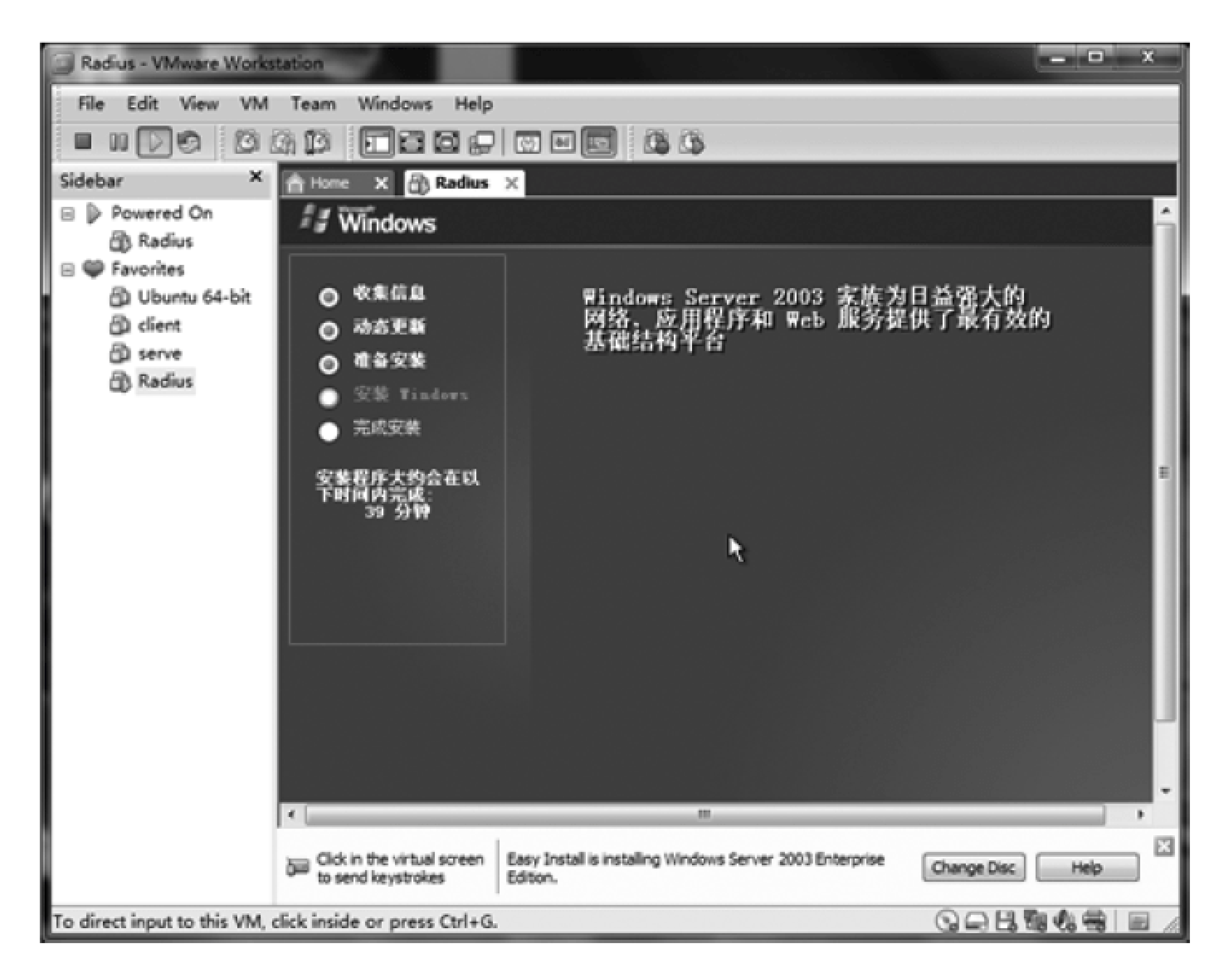


图 1-8　系统安装界面

（3）登入到系统中，在已经搭建好的 Windows Server 2003 平台上进行配置，如图 1-9 所示。



图 1-9　安装完成

（4）登入到系统，在 Windows Server 2003 中安装 VMWare Tools。

【实验报告】

（1）在 VMWare Workstation 7 中安装 Windows Server 2003 过程；

（2）安装 VMWare Tools 软件包的过程。

1.4　小结

通过本实验，将对 VMware 虚拟网络的原理运行机制有一定的了解；并熟悉 VMware 虚拟机的安装；同时熟悉 VMware 虚拟机的三种模式；并学会 VMware 虚拟机中 WireShark 网络数据包的抓取。

参考文献

[1]　姚军. VMware 网络技术：原理与实践[M]. 北京：机械工业出版社，2014.

[2]　徐炯. VMware vSphere 部署的管理和优化[M]. 北京：机械工业出版社，2013.

[3]　王春海，刘晓辉，白凤涛. Vmware 虚拟机实用宝典[M]. 北京：中国铁道出版社，2007.

[4]　王淑江. Windows Server 2012 活动目录管理实践[M]. 北京：人民邮电出版社，2014.

[5]　高晓飞. 网络服务器配置与管理：Windows Server 2003 平台[M]. 北京：高等教育出版社，2009.

[6]　高升. Windows Server 2003 系统管理[M]. 3 版. 北京：清华大学出版社，2010.

第 2 章 IPSec 基础实验

早期 TCP/IP 协议存在种种安全问题，为此 IETF 设计了一套提供 Internet 安全通信的协议，称为 IPSec 协议。该协议在 IP 层提供安全服务，包括机密性、完整性以及认证性。它既可以和 IPv4 联合使用，也可以和 IPv6 联合使用，是互联网的基础安全协议。因此，了解和应用 IPSec 协议也必将成为网络安全课程中的重要内容。

在建立 IPSec 安全通道之前，对等体之间需要相互认证以确定身份。Windows 2003 IPSec 支持三种身份认证：Kerberos、证书和预共享密钥。其中，预共享密钥是最简单的方式。而且，如果终结点不在同一个域中，并且无法获得证书，则预共享密钥是唯一的身份认证选择。本基础实验通过预共享密钥来实现 IPSec 身份认证，用它来展示实验环境和通用步骤。同时为基于 Kerberos 或基于证书的 IPSec 综合实验做好准备。

2.1 IPSec 简介

传统的 IP 数据包具有不安全性，例如可以修改源地址和目标地址；可以查看、修改、删除数据包的内容，还可以发起数据包重放攻击。而 IPSec 为 IP 层提供安全服务，包括机密性、完整性、认证性和密钥管理。由于 IPSec 独立于加密算法，即使加密算法改变了或增加新的算法，也不对其他部分的实现产生影响。另外，IPSec 还可以实现多种安全策略，这样就能避免给不使用该体制的系统成不利影响。

IPSec 能提供的安全服务集包括访问控制、无连接的完整性、数据源认证、拒绝重发包(部分序列完整性形式)、保密性和有限传输流保密性。因为这些服务均在 IP 层提供，所以任何高层协议均能使用它们，例如 TCP、UDP、ICMP、BGP 等。

这些目标是通过使用两大传输安全协议，认证头部(AH)和封装安全负载(ESP)，以及密钥管理程序和协议的使用来完成的。所需的 IPSec 协议集内容及其使用的方式是由用户、应用程序、和/或站点、组织对安全和系统的需求来决定。

2.1.1 IPSec 体系结构

IETF(Internet Engineering Task Force，工程任务组)中的 Security Protocol Working 工作组为 IPSec 制定了如下目标，所制定的 RFC 文档见表 2-1，文档之间的关系见图 2-1。

IPSec 目标：

(1) 该体制不仅适用于 IPv4，也适用于 IPv6；

(2) 可为运行于 IP 顶部的任何一种协议提供保护；

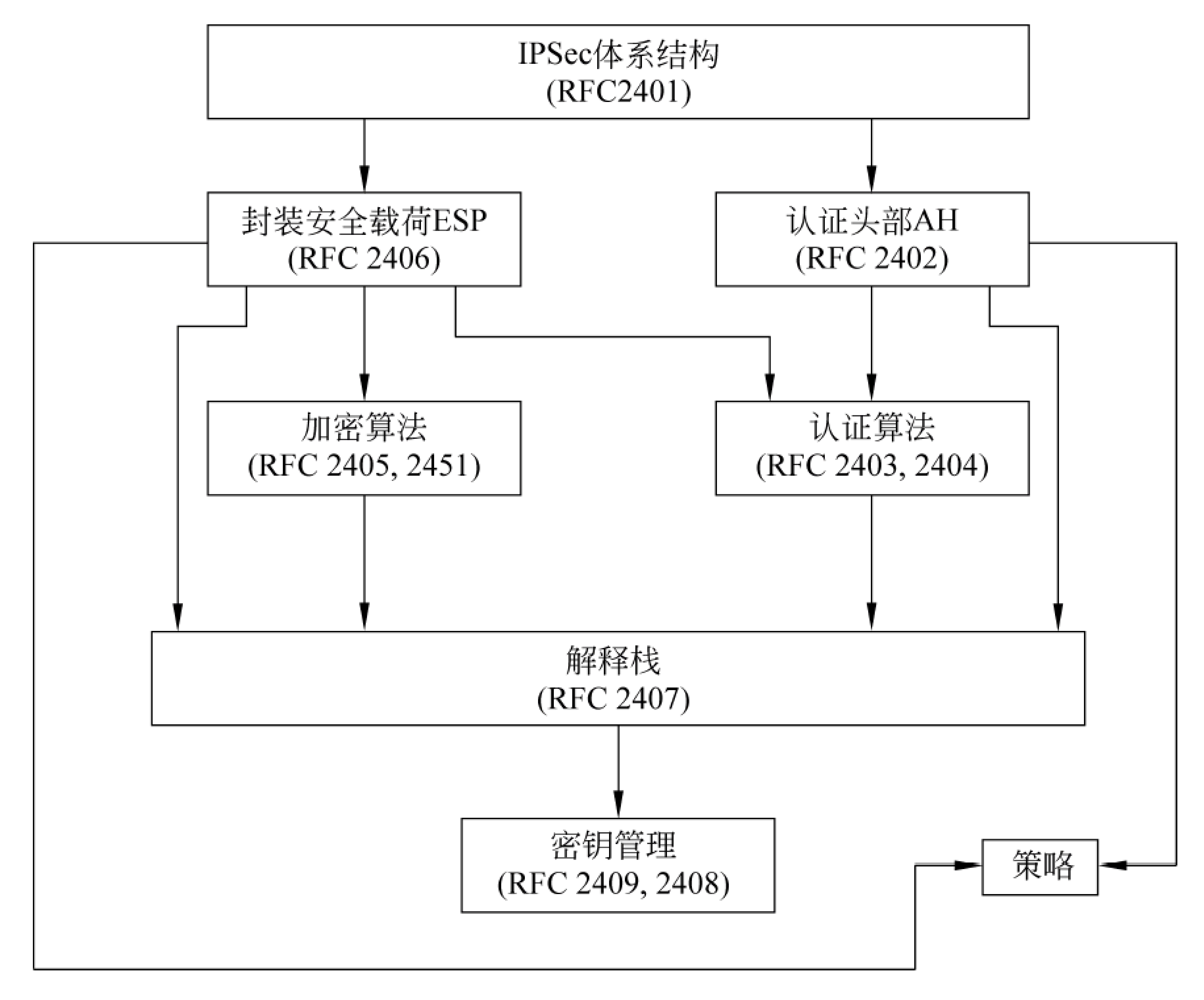


图 2-1 IPSec 文档关系

(3) 即使加密算法改变了或增加新的算法,也不对其他部分的实现产生影响;

(4) 必须能实现多种安全策略,也要避免给不使用该体制的系统成不利影响。

表 2-1 IPSec 文档关系

RFC 编号	RFC 名称	现 状
1825	IPSec 安全体系	RFC2401
1826	IP 认证头部(AH)	RFC2402
18211	IP 封装安全载荷(ESP)	RFC2406
1828	使用 MD5 的 IP 认证	有效
1829	The ESP DES-CBC Transform	有效
2405,2451	加密算法	有效
2403,2404	认证算法	有效
2407	解释域	有效
2409,2408	密钥管理	有效

2.1.2 IPSec 优点

(1) 在路由器或防火墙上执行了 IPSec,就可以为周边的通信提供强有力的安全保障。该防火墙或者路由器所管辖的网络(如一个公司或工作组内部)的通信将不涉及与安全相关的费用。

(2) IPSec 在传输层之下,对于应用程序来说是透明的。当在路由器或防火墙上实现

IPSec 时，无须更改用户或服务器系统中的软件设置。即使在终端系统中执行 IPSec，应用程序一类的上层软件也不会被影响。

(3) IPSec 对终端用户来说是透明的，因此不必对用户进行安全机制的培训。

(4) 如果需要，IPSec 可以为个体用户提供安全保障，这样做可以保护企业内部的敏感信息。

2.1.3 IPSec 工作模式

1. 传输模式

传输模式保护的是 IP 载荷(端到端)，如图 2-2 所示。

IP头	AH/ESP	TCP头	数据	ESP尾部	ESP验证数据
		IP载荷			

图 2-2　IPSec 传输模式

2. 隧道模式

隧道模式保护的是整个 IP 包(网关到网关)，如图 2-3 所示。

新IP头	ESP头部	IP头	TCP头	数据	ESP尾部	ESP验证数据
			IP载荷			

图 2-3　IPSec 隧道模式

2.1.4 AH

AH 的头部信息如图 2-4 所示。

下一个头 (8b)	载荷长度 (8b)	保留字段 (16b)
安全参数索引SPI(32b)		
序列号(32b)		
认证数据(32b的整数倍)		

图 2-4　AH 头部

1. 传输模式下的 AH 认证工作原理

Host_A 发送 | IP头部 | 负载 |，通过 VPN 网关，IPSec 核心处理以后变成 | IP头部 | AH头 | 负载 |在 Internet 上传播，进入到接收方的 VPN 网关，然后经过 IPSec 核心处理，又变回来| IP头部 | 负载 |最后发送给 Host_B。

2. 隧道模式下的 AH 认证工作原理

Host_A → | IP头部 | 负载 | → VPN 网关 → IPSec 核心处理以后 →

[新IP头部][AH头部][IP头部][负载] → Internet → VPN 网关 → IPSec 核心处理 → [IP头部][负载] → Host_B。

2.1.5　ESP

ESP 头部信息如图 2-5 所示。



安全参数SPI(32 b)
序列号(32 b)
载荷数据

图 2-5　ESP 头部

ESP 尾部信息如图 2-6 所示。

填充位	填充长度	下一个首部
ESP认证数据		

图 2-6　ESP 尾部

1. 传输模式下的 ESP 工作原理

Host_A 首先被加上 IP 头部以后，形成 [IP头部][负载]，然后被送往 VPN 网关，经过 IPSec 核心处理以后，形成结构 [IP头部][ESP头部][负载][ESP尾部][ESP认证信息]，再经过 Internet 传输送往接收方 VPN 网关，同样经过 IPSec 核心处理之后，得到 [IP头部][负载]，最后发送给 Host_B。

2. 隧道模式下的 ESP 工作原理

Host_A 首先被加上 IP 头部，形成 [IP头部][负载]，经过发送方 VPN 网关然后经过 IPSec 核心处理以后形成 [新IP头部][ESP头部][IP头部][负载][ESP尾部][ESP认证信息] 的结构，然后被 Internet 传输，到达接收方 VPN 网关，经过 IPSec 核心处理以后，还原成 [IP头部][负载]，最后到达 Host_B。

2.1.6　Internet 密钥交换

因特网密钥交换协议(Internet Key Exchange，IKE)提供自动建立安全关联和管理密钥的功能，它分为 3 部分，包括两个阶段。

(1) ISAKMP：该框架定义了安全关联的管理和密钥管理，以及用于交换密钥产生和认证数据的报文负载，它本身没有定义任何密钥交换协议。

(2) Oakley：是一个可用于 ISAKMP 框架的密钥交换协议，它为安全关联提供密钥交换和刷新功能。

(3) SKEME：安全密钥交换机制。

阶段 1：主要用来建立对 ISAKMP 消息自身的保护措施，它并不建立用于保护用户

数据流的安全关联或密钥。同时,协商建立一个主秘钥,以后用于保护用户数据流的所有秘钥都将根据主密钥产生。

阶段 2:协商建立安全关联和将用于保护用户数据流的密钥。

2.2 预共享密钥的 IPSec 实验

【实验目的】

(1) 了解 IPSec 的原理和协议运行机制;

(2) 掌握 IPSec 身份认证的预共享密钥的配置;

(3) 掌握用 WireShark 工具抓包分析 IPSec 数据包格式和协议流程。

【实验内容】

用户 A 的计算机名为 Alice,IP 地址为 192.168.x.x1;用户 B 的计算机名为 Bob,IP 地址为 192.168.x.x2;它们之间的 ICMP 通信需要使用 IPSec 来保证机密性和完整性,其身份认证方式为预共享密钥。

【实验环境】

安装 Windows 操作系统的 PC1 台,并在控制台 MMC 中,添加“IP 安全策略”和“IPSec 监视器”两个单元,以配置 IP 安全策略以及观察 IPSec 通信状态。

【实验参考步骤】

(1) 使用两台虚拟机 NQP1 和 NQP2 进行实验,如图 2-7 所示。

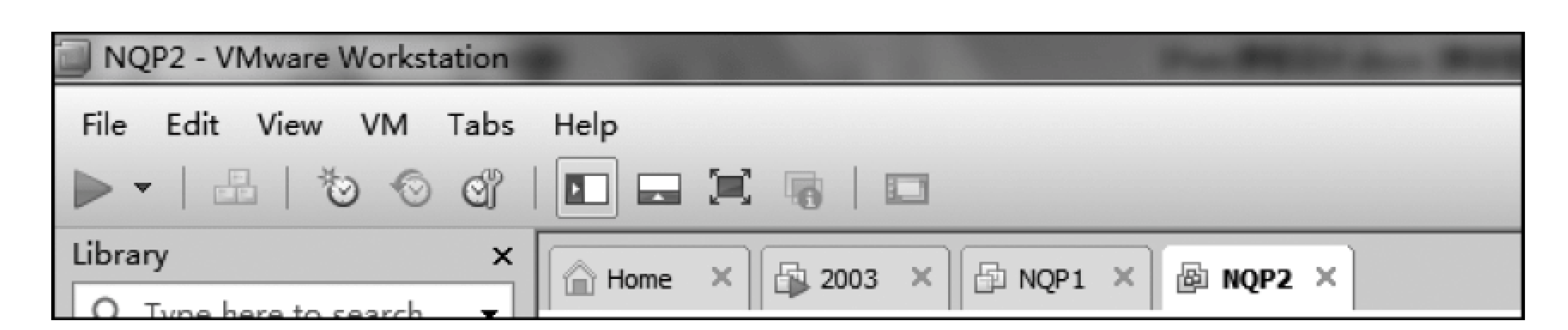


图 2-7 实验用的虚拟机

(2) 初始状态验证两台虚拟机能够相互通信,如图 2-8 所示。

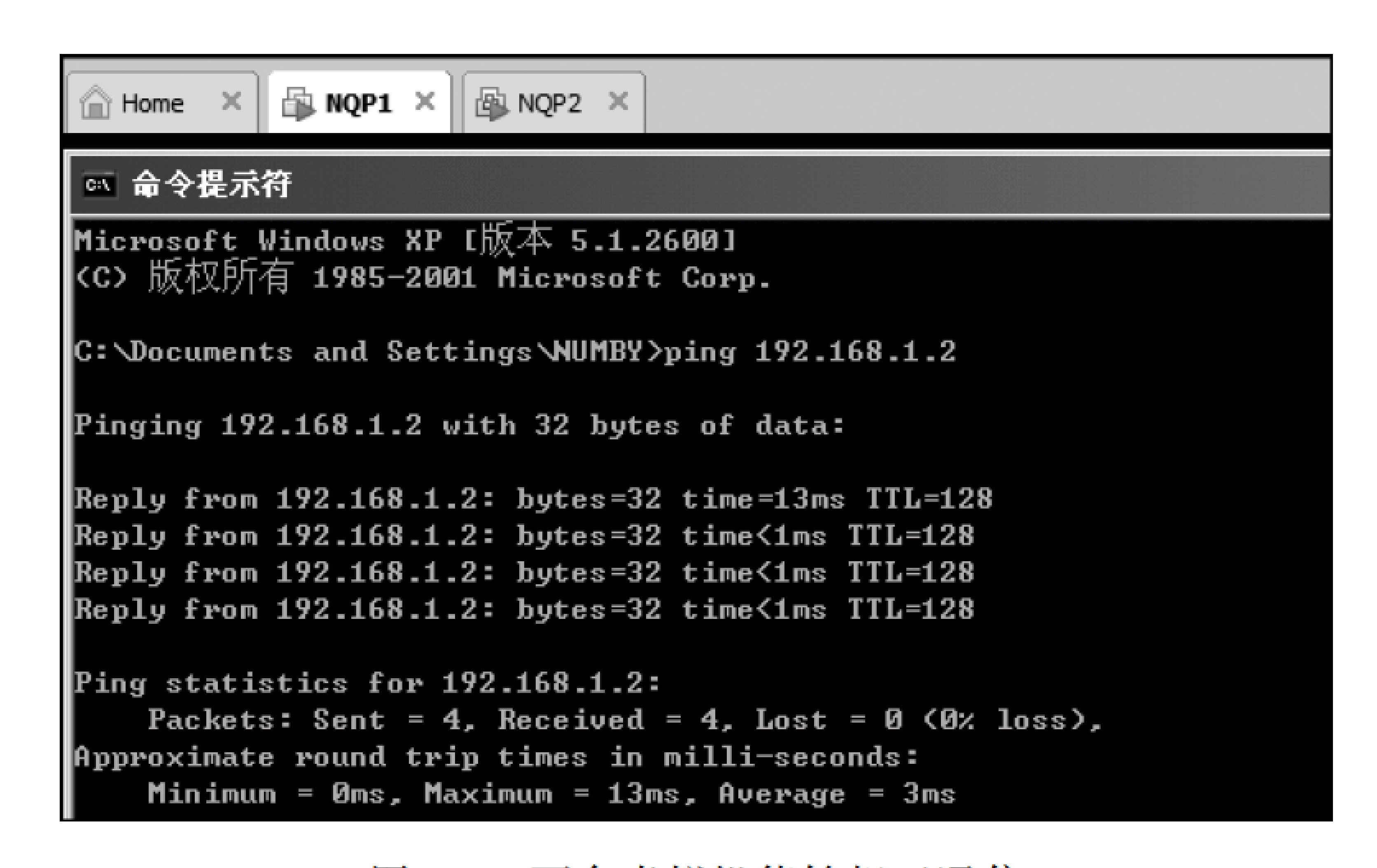


图 2-8 两台虚拟机能够相互通信

(3) 对第一台虚拟机进行安全策略配置，如图 2-9 所示。

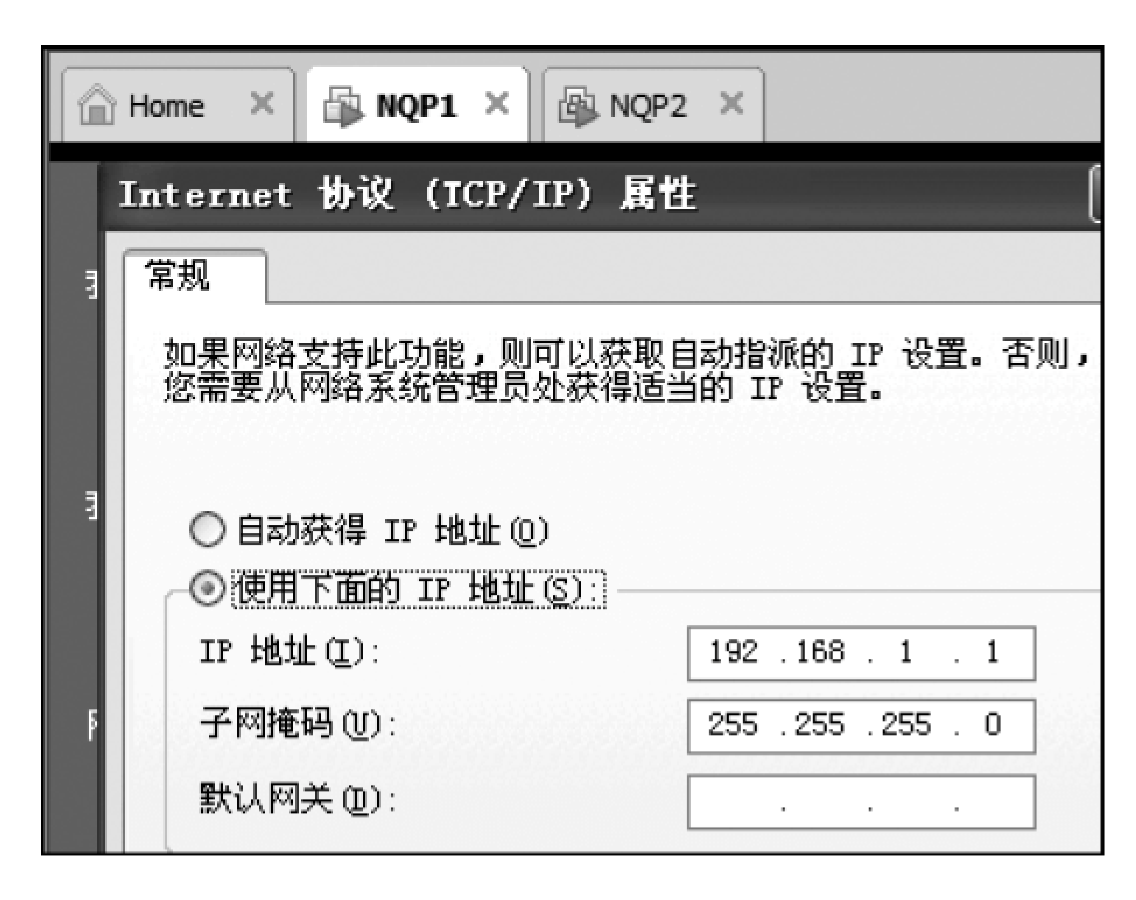


图 2-9　第一台虚拟机的安全策略配置

(4) 安全策略的筛选器操作，如图 2-10 所示。



图 2-10　安全策略的筛选器操作

(5) 预共享密钥，如图 2-11 所示。

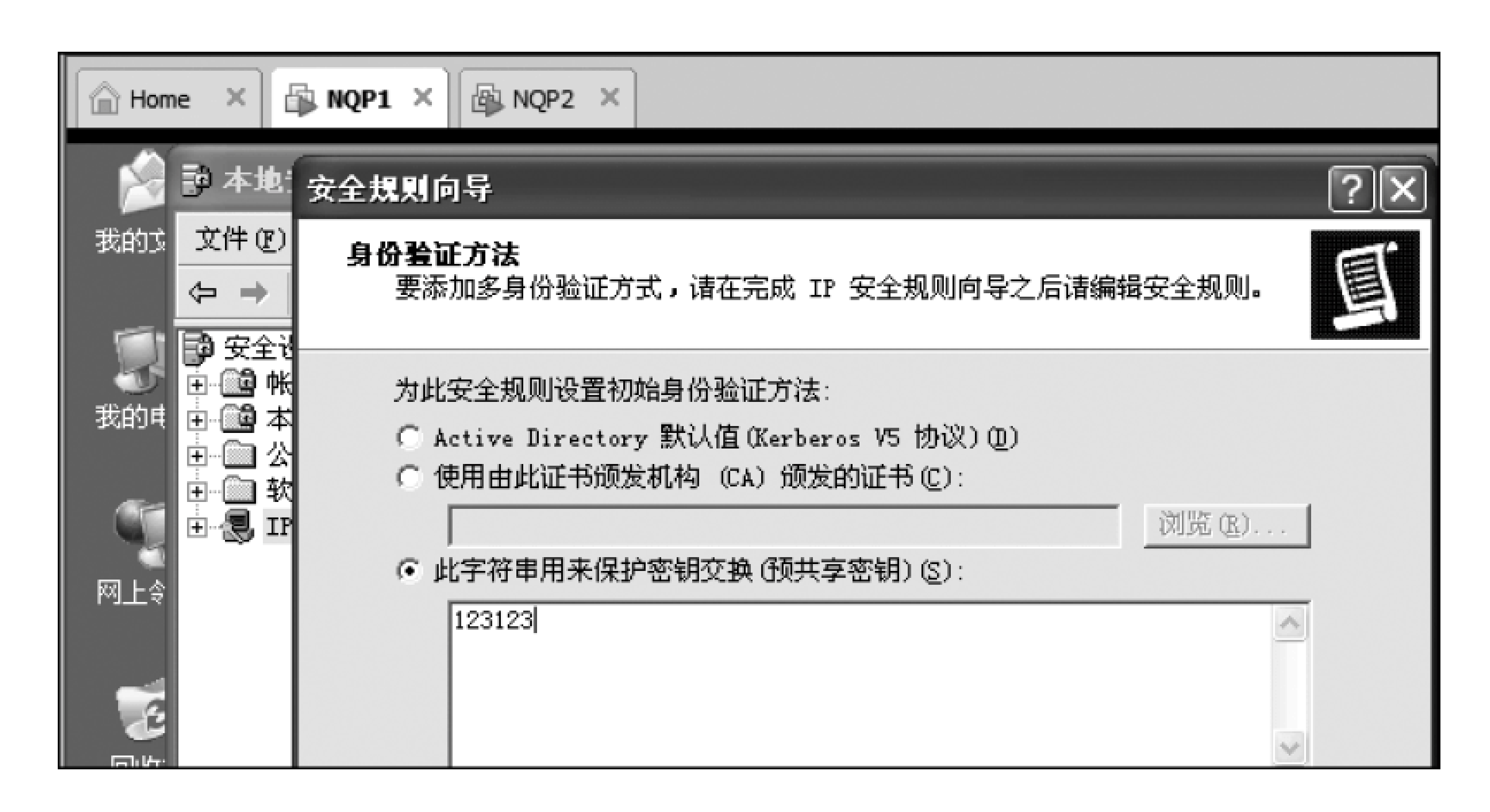


图 2-11　预共享密钥

(6) 对第二台虚拟机进行安全策略配置，如图 2-12 所示。

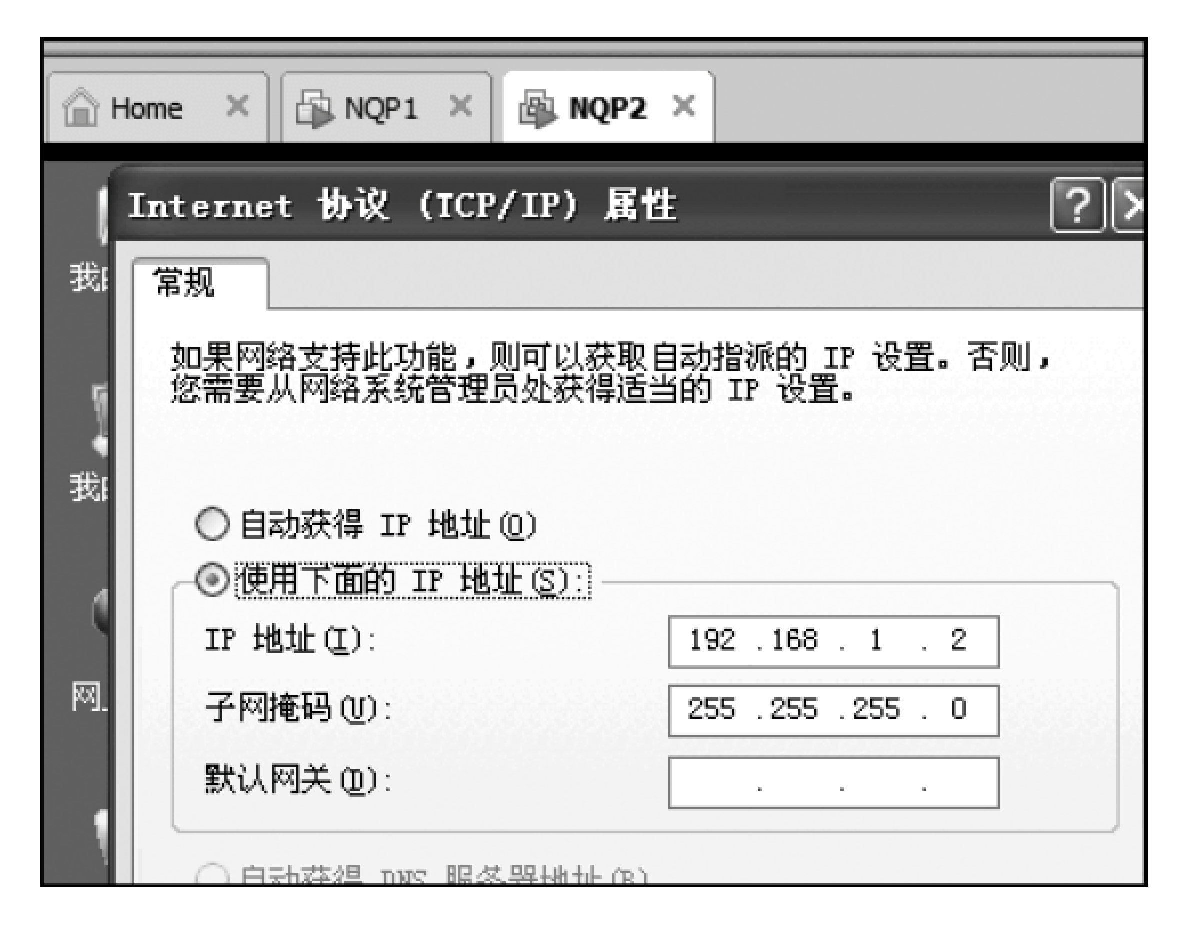


图 2-12　第二台虚拟机的安全策略配置

(7) 安全策略的筛选器操作，如图 2-13 所示。

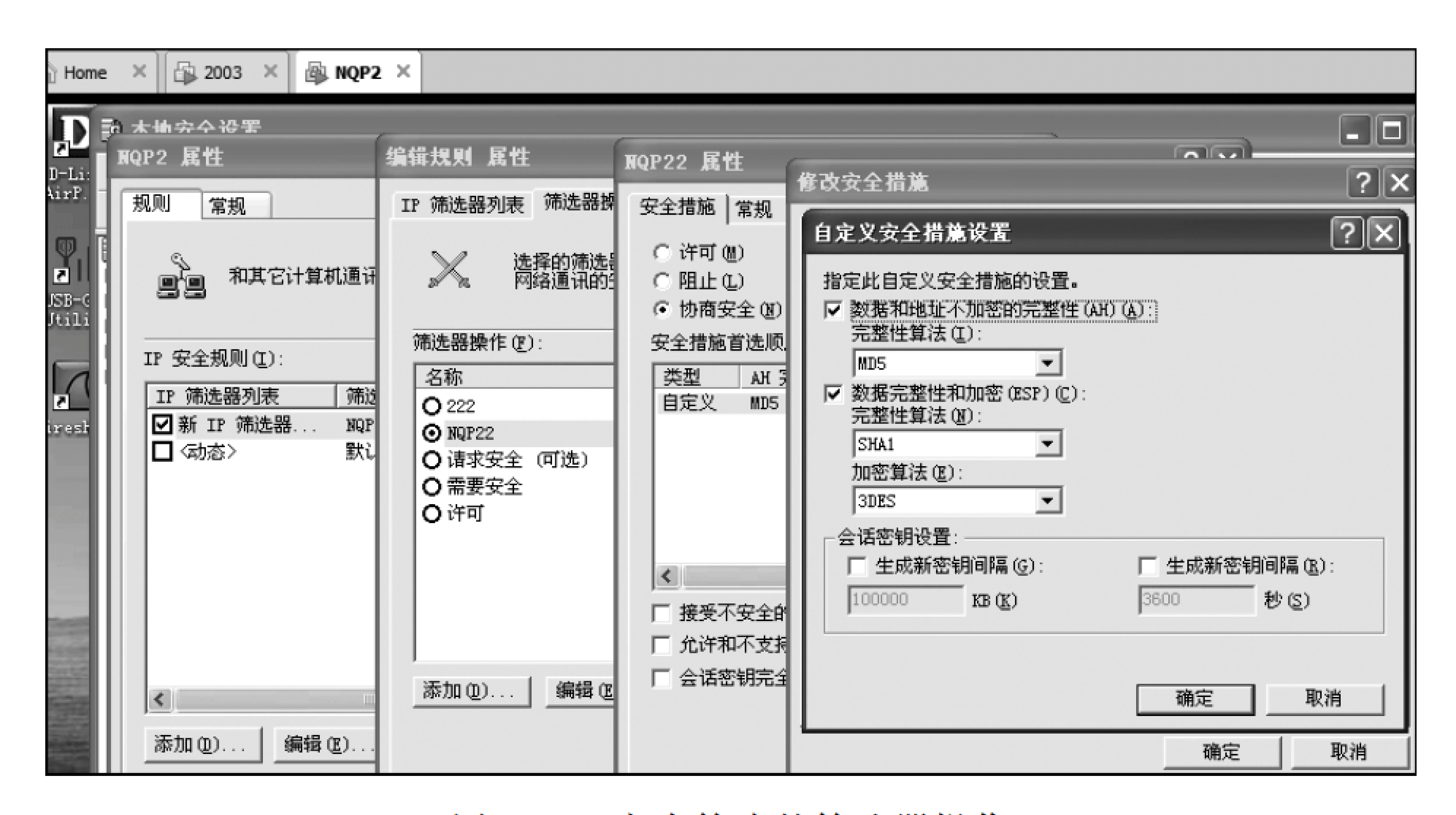


图 2-13　安全策略的筛选器操作

(8) 预共享密钥，如图 2-14 所示。

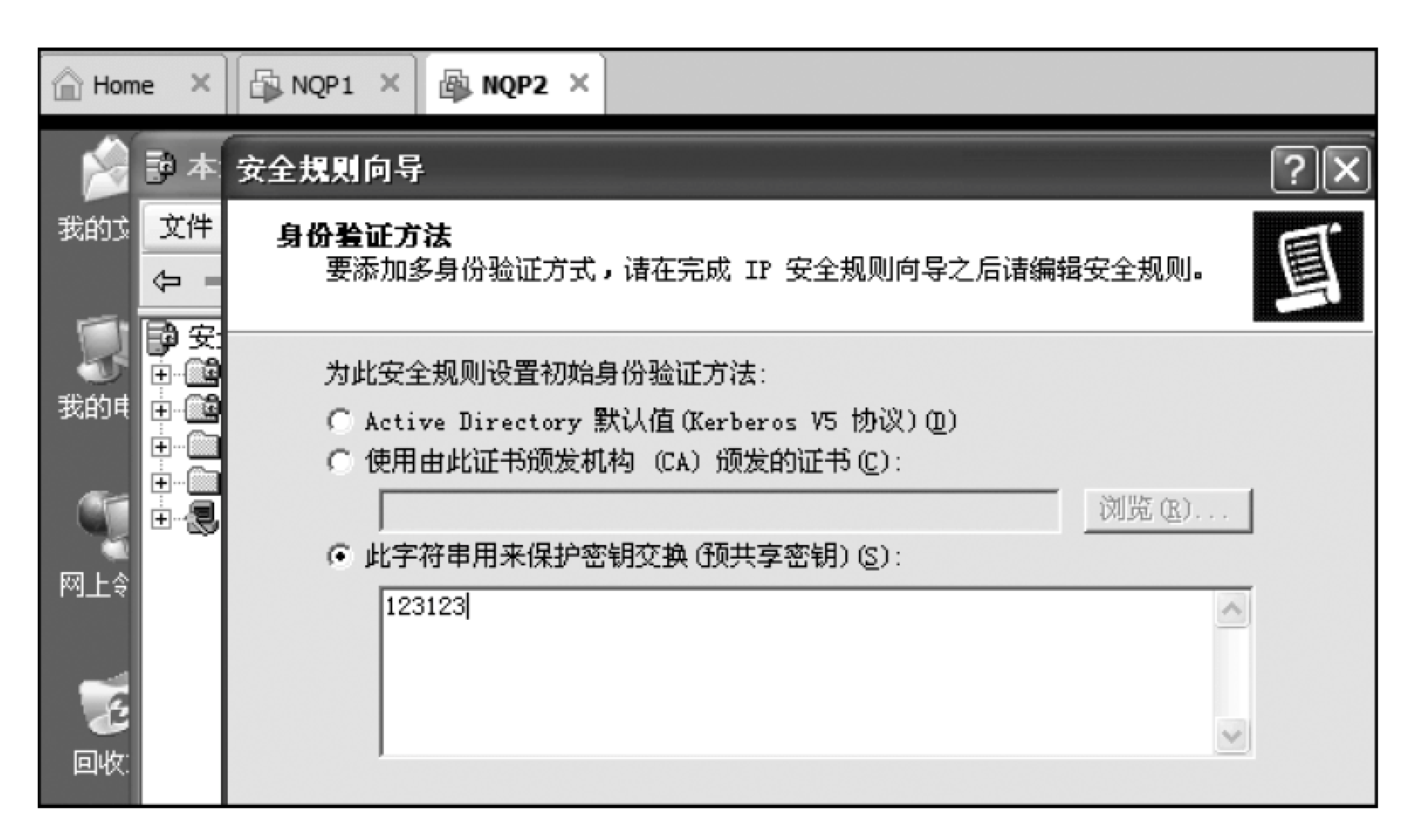


图 2-14　预共享密钥

由此看出，两台虚拟机所配置的安全策略一样。

(9) 当两台虚拟机同时指派策略时，则可以相互 ping 通，如图 2-15 所示。

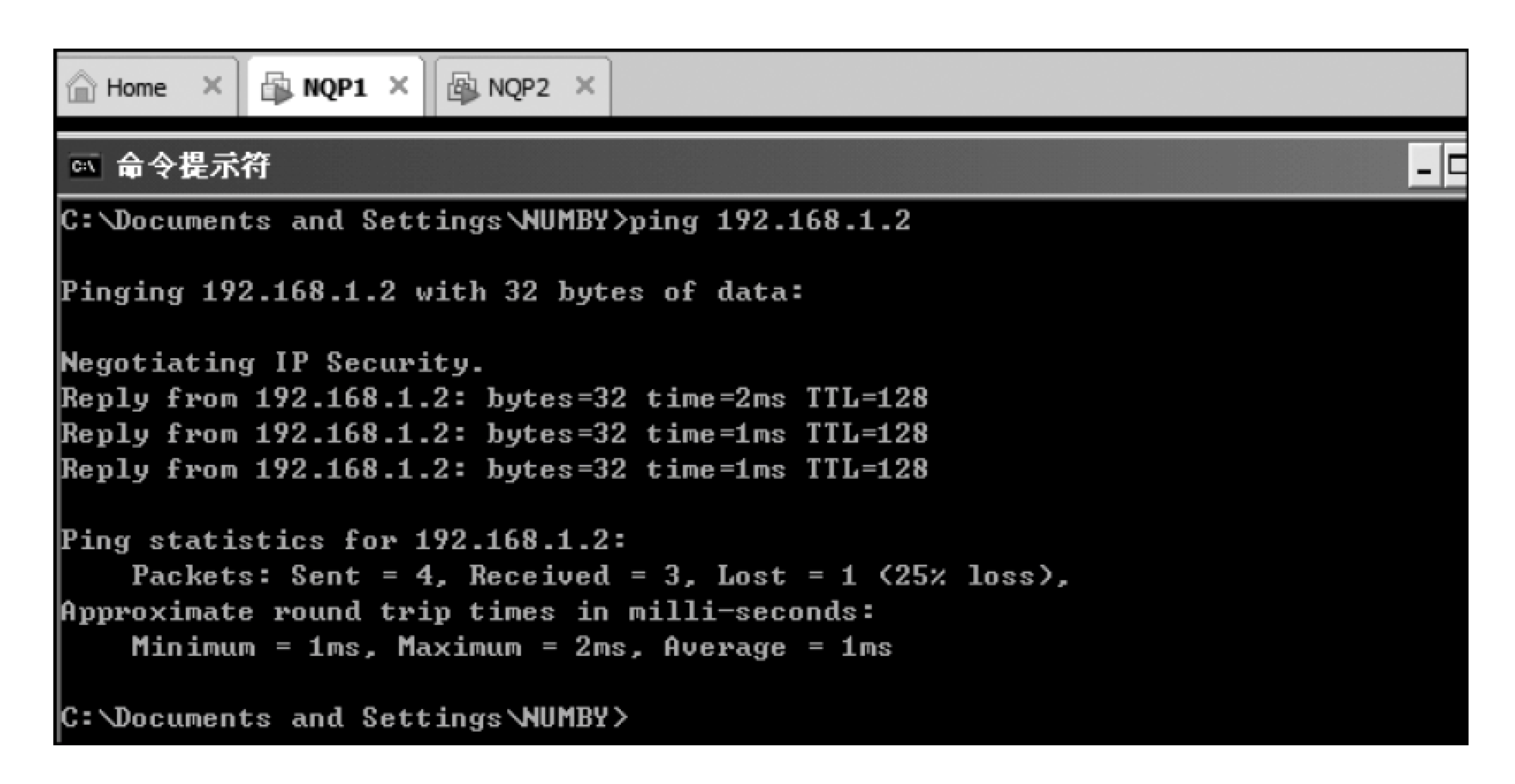


图 2-15　ping 结果

(10) 使用抓包工具 WireShark 抓包，如图 2-16 所示。

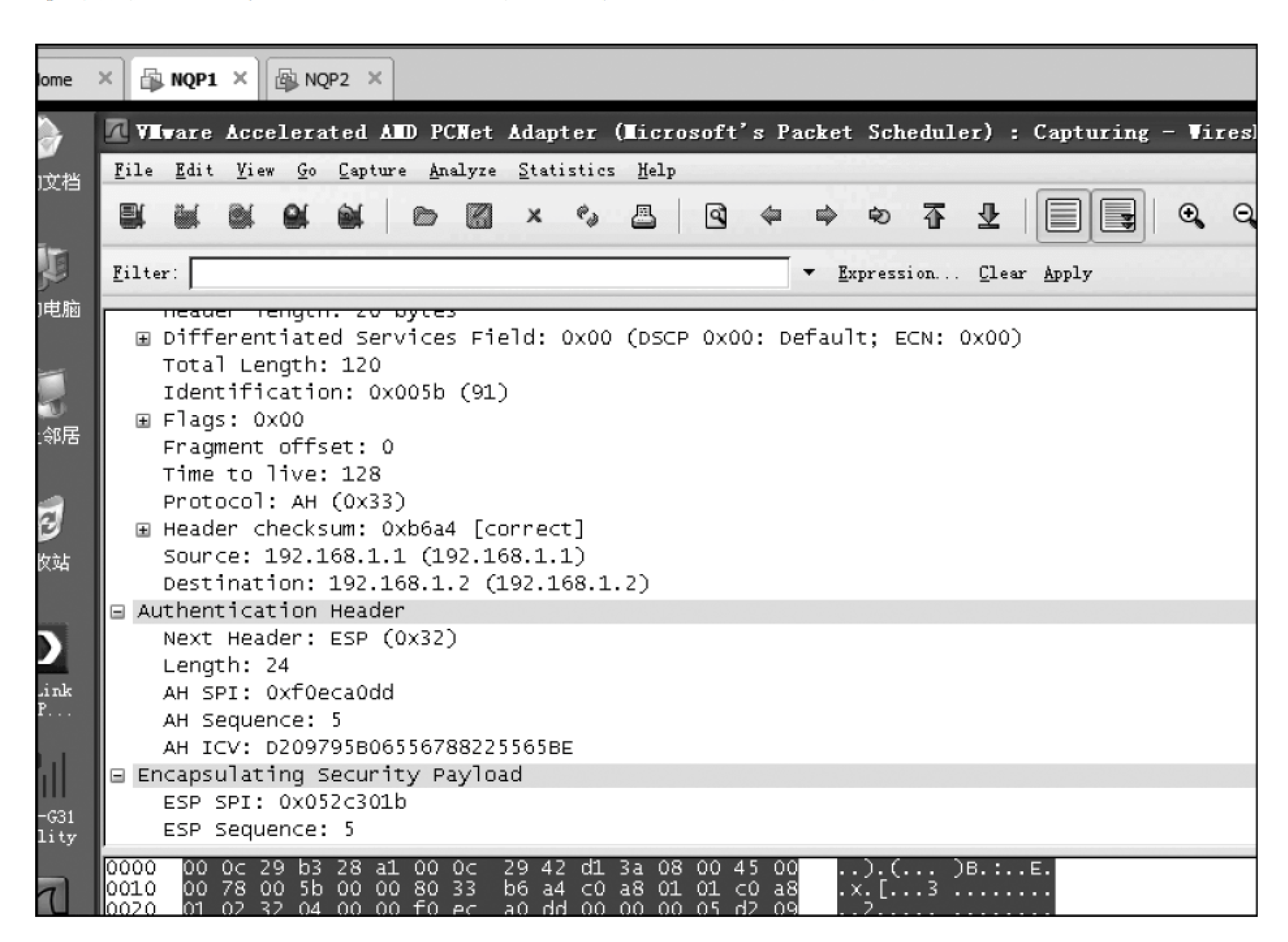


图 2-16　WireShark 抓包

其中含有 AH 头部和 ESP 头部。

(11) 查看主关联，如图 2-17 所示。

【实验报告】

(1) IPSec 的原理和协议运行机制；

(2) IPSec 身份认证的预共享密钥的配置过程；

(3) WireShark 工具抓包分析 IPSec 数据包格式和协议流程。

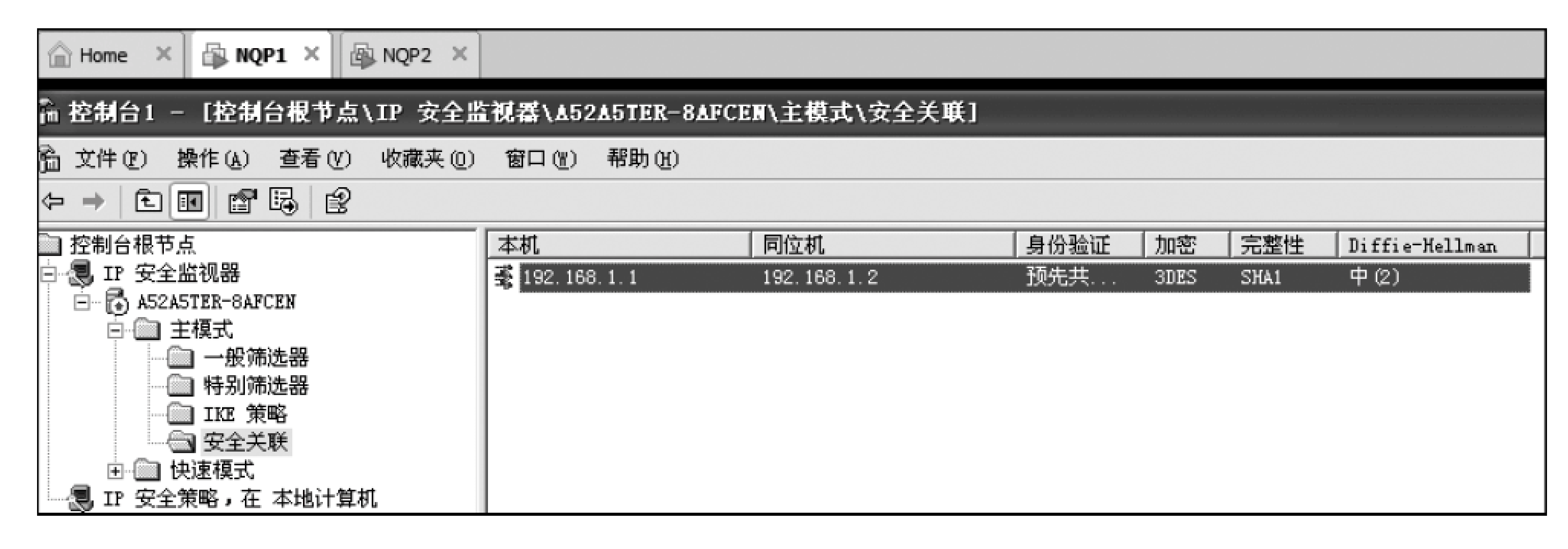


图 2-17　主关联

2.3　小结

通过本实验，将对 IPSec 的原理和协议运行机制有一定的了解；并熟悉 IPSec 数据包格式和协议流程；更重要的，熟悉 IPSec 身份认证的实验环境和通用步骤；为后面的基于 Kerberos 或基于证书的 IPSec 综合实验做好准备。

参考文献

[1] S. Wainner. IPSec VPN 设计[M]. 袁国忠译. 北京：人民邮电出版社，2012.
[2] 秦柯. Cisco IPSec VPN 实战指南[M]. 北京：人民邮电出版社，2012.
[3] 周世杰，陈伟，罗绪成. 计算机系统与网络安全技术[M]. 北京：高等教育出版社，2011.
[4] 王志海等. OpenSSL 与网络信息安全：基础、结构和指令[M]. 北京：清华大学出版社，2007.
[5] 杨浩淼，张文科，蒋磊. 基于 VMWare 的 IPSec 综合实验设计[J]. 通信技术，2012(6).
[6] 王梦龙. 网络信息安全原理与技术[M]. 北京：中国铁道出版社，2009.
[7] 吴迪，何坚，潘嵘，刘聪. 基于 VMware 虚拟网络的 IPSec 实验教学[J]. 实验技术管理，2010(9).
[8] Doraswamy，Naganand，Harkins，Dan. IPSec：The New Security Standard for the Internet，Intranet，and Virtual Private Networks[M]. Upper Saddle River：Prentice Hall，2003.

第 3 章 SSL 基础实验

SSL(Secure Sockets Layer,安全套接层)协议是网景公司提出的基于 Web 应用的安全协议。SSL 协议指定了一种在应用层协议和可靠的传输层协议(如 TCP 协议)之间提供数据安全性分层的机制。它可以为网络连接提供数据加密、服务器认证、消息完整性以及可选的客户端认证。

本章要求是进行 SSL 相关实验,具体内容包括在 VMware 环境下实现 SSL 配置、SSL+FTP 服务器搭建以及 http 与 https 通信数据分析对比。

3.1 SSL 简介

SSL 是一种在两台机器之间提供安全通道的协议,它具有保护传输数据以及识别通信机器的功能。安全通道是透明的,意思就是说它对传输的数据不加变更。客户与服务器之间的数据是经过加密的,一端读取的数据完全是另一端写入的内容。透明性使得几乎所有基于 TCP 的协议稍加改动就可以在 SSL 上运行,非常方便。

SSL 原先是为 Web 设计的。网景公司的意图是针对其所有的通信安全问题,包括 Web、电子邮件及新闻组通信提供一种单一的解决方案。SSL 经过了多次修改,从开始的版本 1 到 IETF 最终所采纳的 TLS 标准。版本 1 从未经过广泛的部署,所以一般都认为从版本 2 开始。图 3-1 描述了各种 SSL 变种的谱系树。

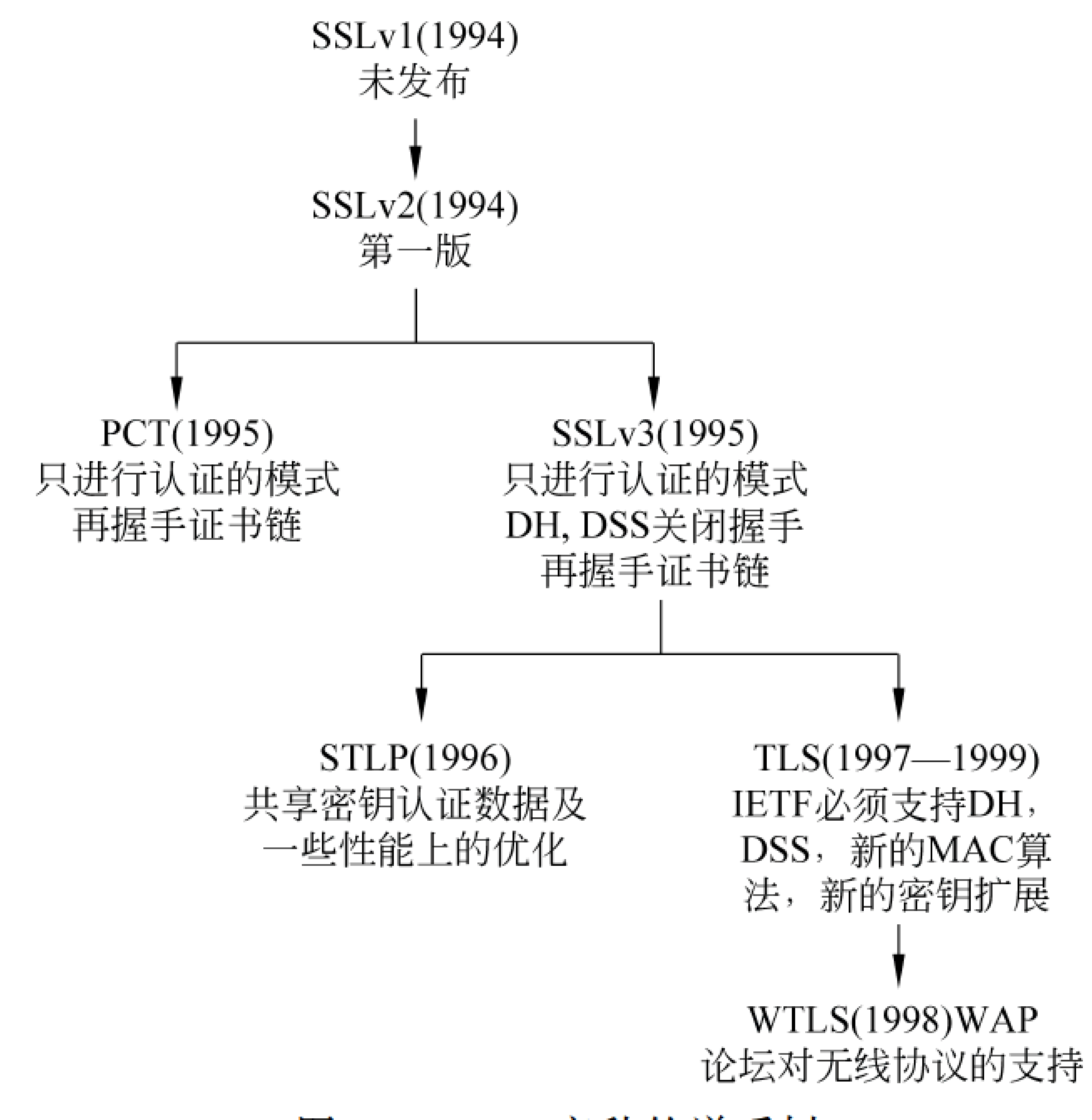


图 3-1　SSL 变种的谱系树

SSL 假定其下层的数据包发送机制是可靠的。写入网络的数据将依顺序发送给另一端的程序，不会出现丢包或重复发送的情况。从理论上讲，有许多传输协议都能提供此种服务，但在实际应用中，SSL 几乎只是在 TCP 上运行，它不能在 UDP 或直接在 IP 上运行。

3.1.1 SSL 协议结构

SSL 协议是一种分层协议，分为记录层协议和位于其上的握手层协议。握手层协议又可以细分为四种协议：握手协议（Handshake Protocol）、修改密钥参数协议（Change Cipher Spec Protocol）、警告协议（Alert Protocol）和应用数据协议（Application data Protocol）。握手层协议需要记录层协议的支持。图 3-2 描述了该协议的结构及其在 TCP/IP 协议栈中的位置。

握手协议	修改密钥参数协议	警告协议	应用数据协议
记录协议			
TCP			
IP			

图 3-2 SSL 协议结构

SSL 连接分为两个阶段，即握手阶段和数据传输阶段。握手阶段对服务器进行认证并确立用于保护数据传输的加密密钥。握手完成之后，才能够开始传输数据。在传输阶段，数据被分成一系列经过保护的记录进行传输。

3.1.2 SSL 握手

SSL 握手有三个目的：第一，客户端与服务器端需要就一组用于保护数据的算法达成一致；第二，它们需要确立一组加密密钥；第三，握手还可以选择对客户端进行认证。在传输任何应用数据之前，都要使用握手协议。握手的整个工作过程如图 3-3 所示。

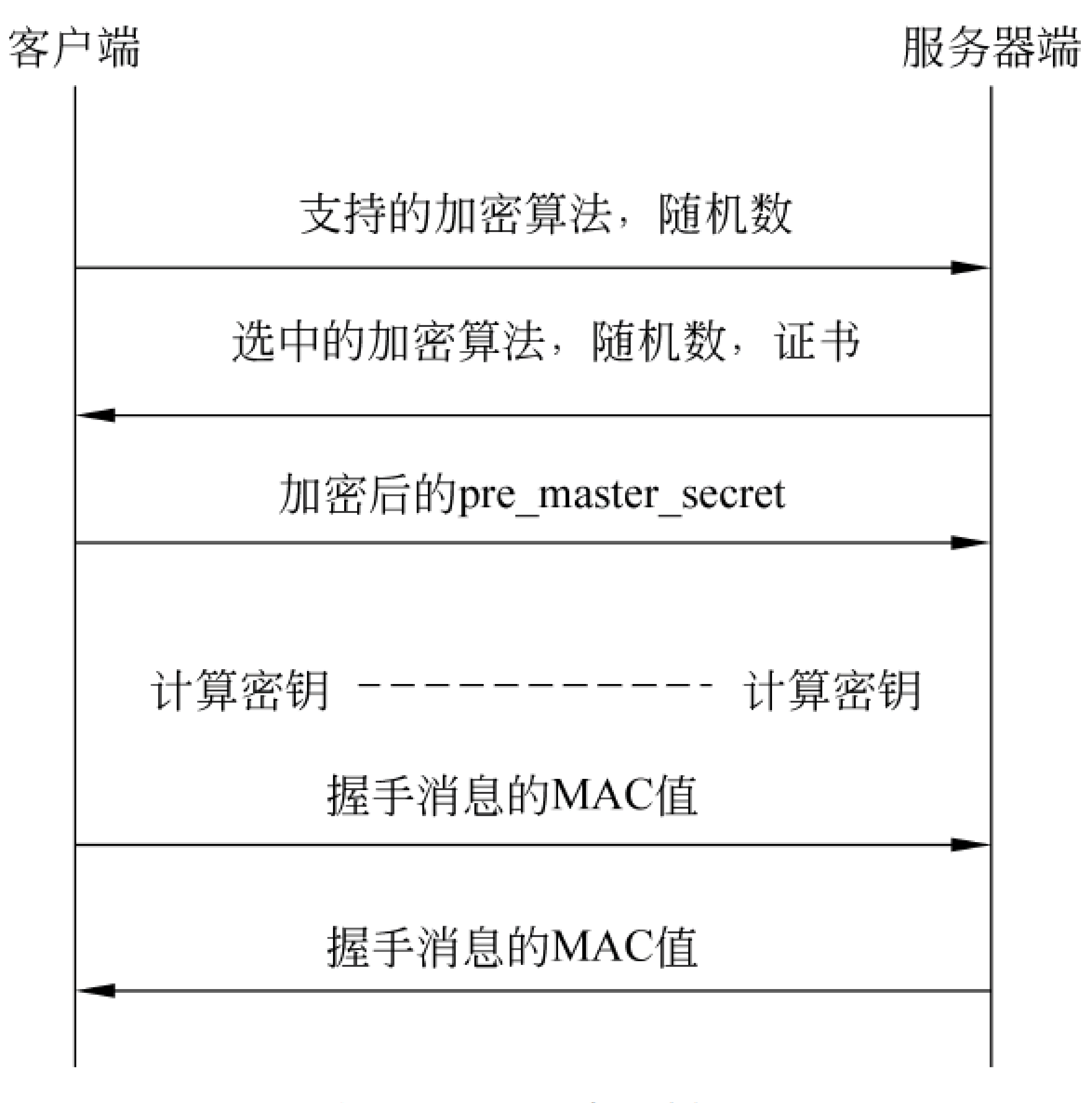


图 3-3 SSL 握手概述

（1）客户端将它所支持的算法列表连同一个密钥产生过程用作输入的随机数发送给服务器。

（2）服务器根据从列表的内容中选择一种加密算法，并将其连同一份包含服务器公钥的证书发回给客户端。该证书还包含了用于认证目的的服务器标识，服务器同时还提供了一个作为密钥产生过程部分输入的随机数。

（3）客户端对服务器的证书进行验证，并抽取服务器的公钥。然后，再产生一个称作 pre_master_secret 的随机密码串，并使用服务器的公钥对其进行加密。最后，客户端将加密后的信息发送给服务器。

（4）客户端与服务器端根据 pre_master_secret 以及客户端与服务器的随机数值独立计算出加密和 MAC（Message Authentication Code，消息认证码）密钥。

（5）客户端将所有握手消息的 MAC 值发送给服务器。

（6）服务器将所有握手消息的 MAC 值发送给客户端。

3.1.3 记录协议

握手的目的是建立起使发送和接收数据都受到保护的安全连接，而实际的数据传输需要使用记录协议来实现。

记录协议是通过将由高层获得的数据分割成一系列的片段并加以传输来工作的，其中对每个片段单独进行保护和传输。在接收方，对每条记录单独进行解密和验证。这种方案使得数据一经准备好就可以从连接的一端传送到另一端，接收端在接收到片段后可以立刻进行处理。

在传输片段前，先对每个片段进行压缩，然后通过计算压缩片段的 MAC 来保证其完整性，其中计算 MAC 使用的散列算法由握手过程协商的结果得到。MAC 与压缩片段一起进行传输，并由接收端加以验证。将 MAC 附加到压缩片段的尾部，然后对整个压缩片段和 MAC 进行加密，得到相应的负载。最后给负载加上头信息，头信息与负载一起就成为一条记录。

记录层协议工作流程如图 3-4 所示。详细步骤如下：

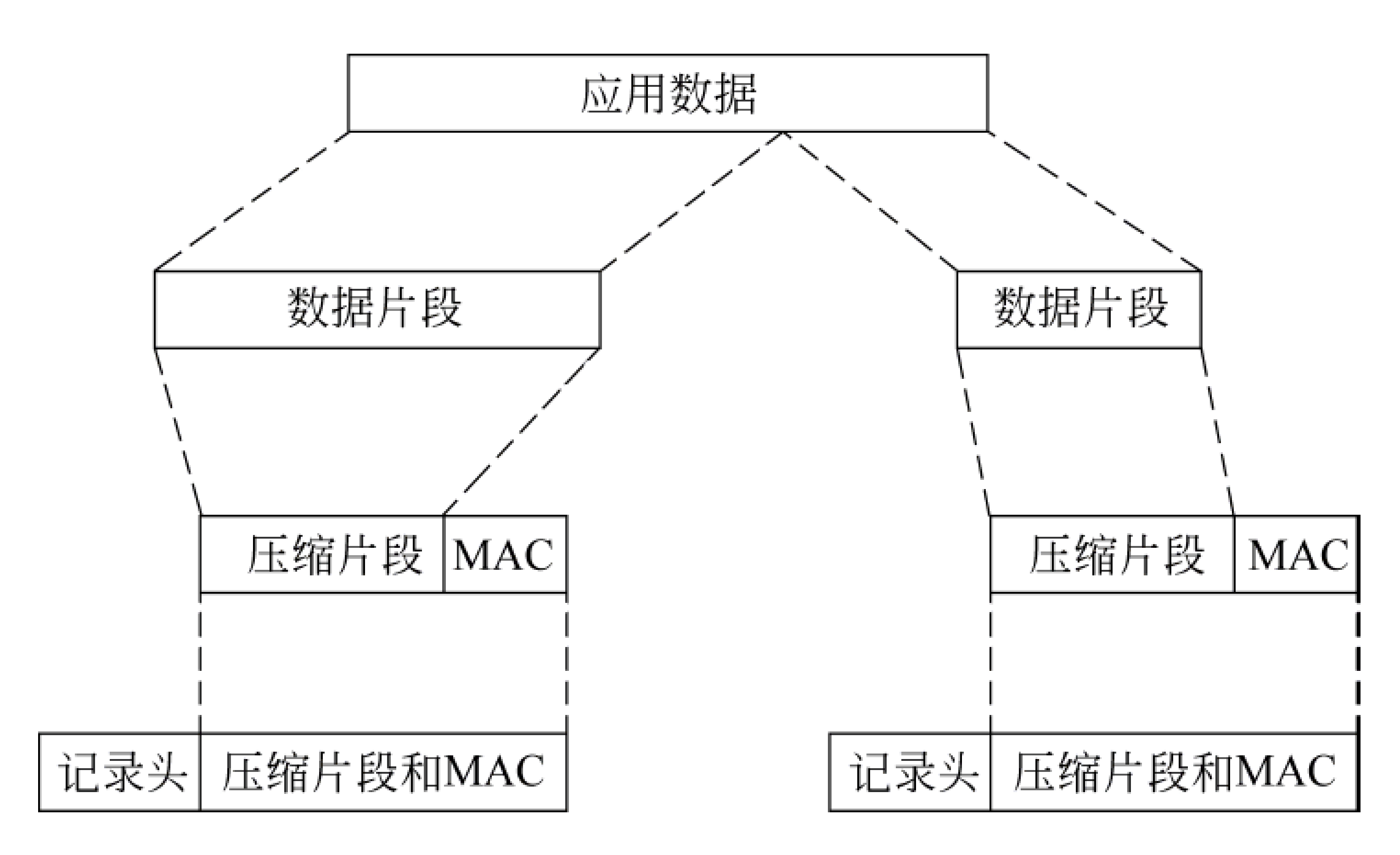


图 3-4　SSL 记录协议的工作流程

（1）将上层数据分成适当大小的块。每个块的大小不得超过 2^{14}B（16 384B）。

(2) 对数据块进行压缩,压缩不能丢失信息,并且增加的长度不能超过1024字节。

(3) 在压缩后的数据上计算MAC。此时需要使用共享的密钥。

(4) 使用同步加密算法对压缩报文加上MAC进行加密。加密对内容长度的增长不得超过1024字节,因此总长度不得超过214+2048字节。

(5) 对于流加密,压缩报文和MAC一起被加密;对于分组加密,在MAC之后,加密之前可以增加填充(padding)。填充由表示填充长度的字节跟着一定数目的填充字节组成,填充字节的数目是使得要加密的数据(明文加上MAC,再加上填充)的总长度成为加密分组长度整数倍的最小数目。例如,一个58字节的明文(如果使用压缩,就是压缩正文),带有20字节的MAC,使用分组长度为8字节的算法进行加密(如DES)。加上填充长度字节,产生的总长度为79字节。为了使得总长度为8的整数倍,需要增加一个字节的填充。

(6) SSL记录协议处理的最后一个步骤是附加一个首部。

3.1.4 SSL警告和修改密钥参数协议

警告协议允许连接的一端向另一端报告警告信息。警告信息包括警告内容和严重级别。如果收到fatal级别的警告消息,则当前连接立即中断,并清除包含会话标识、密钥在内的所有状态参数。在这种情况下,与该会话相连的其他连接可以继续进行,但不能再重用该会话建立新的连接。同其他的消息一样,警告信息也要经过压缩和加密后再进行传输。警告消息的类型可分为关闭警告(close alert)和错误警告(error alert)。

修改密钥参数协议用来标识信号的转换,它只包含一条信息,信息内容为一个字节,其值为1。修改密钥参数消息是由客户方或服务器发出,用来通知对方随后的记录将受到刚协商的密钥参数和密钥的保护。从握手协议的流程图可看出,客户方在发送完密钥交换和证书验证消息后发送修改密钥参数消息,服务器在成功处理了从客户方接收到的密钥交换消息以后也发送一个修改密钥参数消息。如果在握手过程中采用了会话重用,则双方在hello消息之后发送修改密钥参数消息。

3.2 SSL配置实验

【实验目的】

(1) 掌握对Web数据进行SSL安全传输的技术;

(2) 掌握证书申请的过程;

(3) 熟悉X.509 v3证书格式。

【实验内容】

配置基于Web的SSL连接:

(1) IIS服务器的启动;

(2) 证书服务的启动和独立根CA的安装;

(3) 服务器证书申请、颁发和安装;

(4) 客户端证书申请、颁发和安装；

(5) 在服务器上配置 SSL；

(6) 客户端通过 SSL 与服务器建立连接；

(7) WireShark 抓包，分析 SSL 记录，并绘制 SSL 握手的流程。

【实验环境】

安装 Windows XP 的计算机、安装 Windows Server 2003 的计算机、IIS 的 Web 服务器和证书服务、局域网。

【实验参考步骤】

本实验主要讲述 SSL 配置、基于证书的身份认证和 CA 服务器搭建，即使用证书在 Web 服务器上设置 SSL。

(1) 在服务器上安装 IIS 组件，并配置 Web 站点，如图 3-5 至图 3-7 所示。



图 3-5　安装 IIS



图 3-6　IIS 配置



图 3-7 配置 Web 服务器

(2) 使用 IE 浏览器输入 IP 地址访问本地站点,如图 3-8 所示。

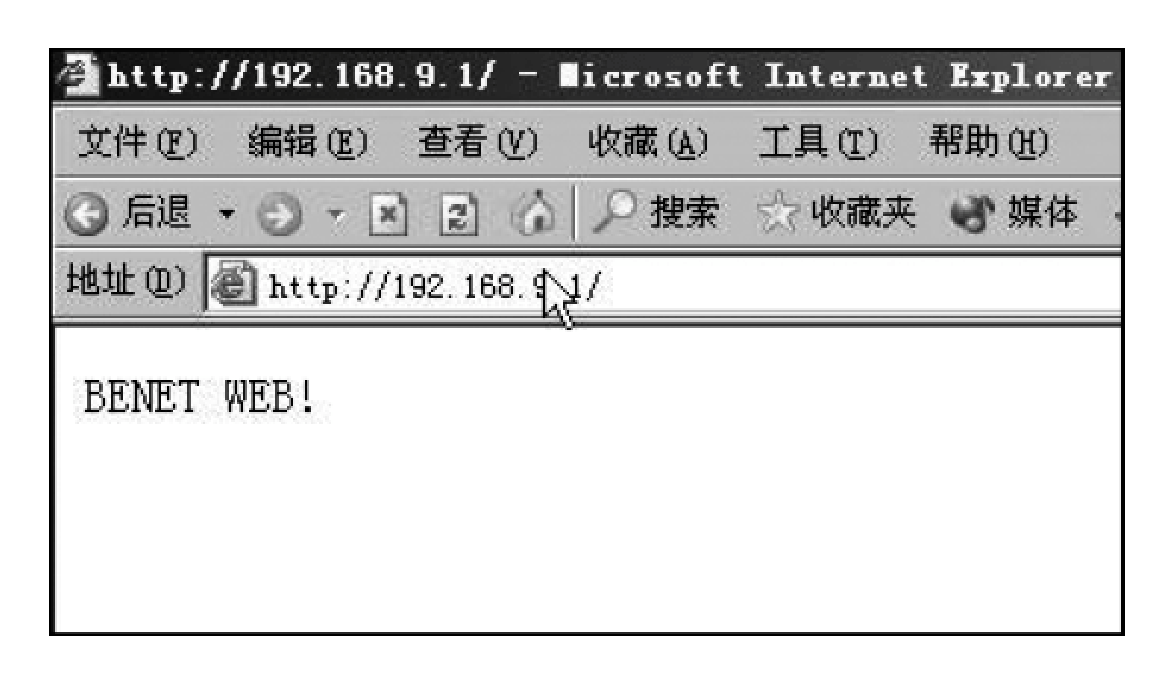


图 3-8 http 访问站点

(3) 在服务器上安装 CA 组件,如图 3-9 所示。



图 3-9 CA 安装

(4) 因为没有 AD(活动目录),所以只能创建独立 CA,又因为是第一个 CA,所以选择独立根 CA,并选择自定义安装,如图 3-10 至图 3-13 所示。



图 3-10　设置 CA 名字

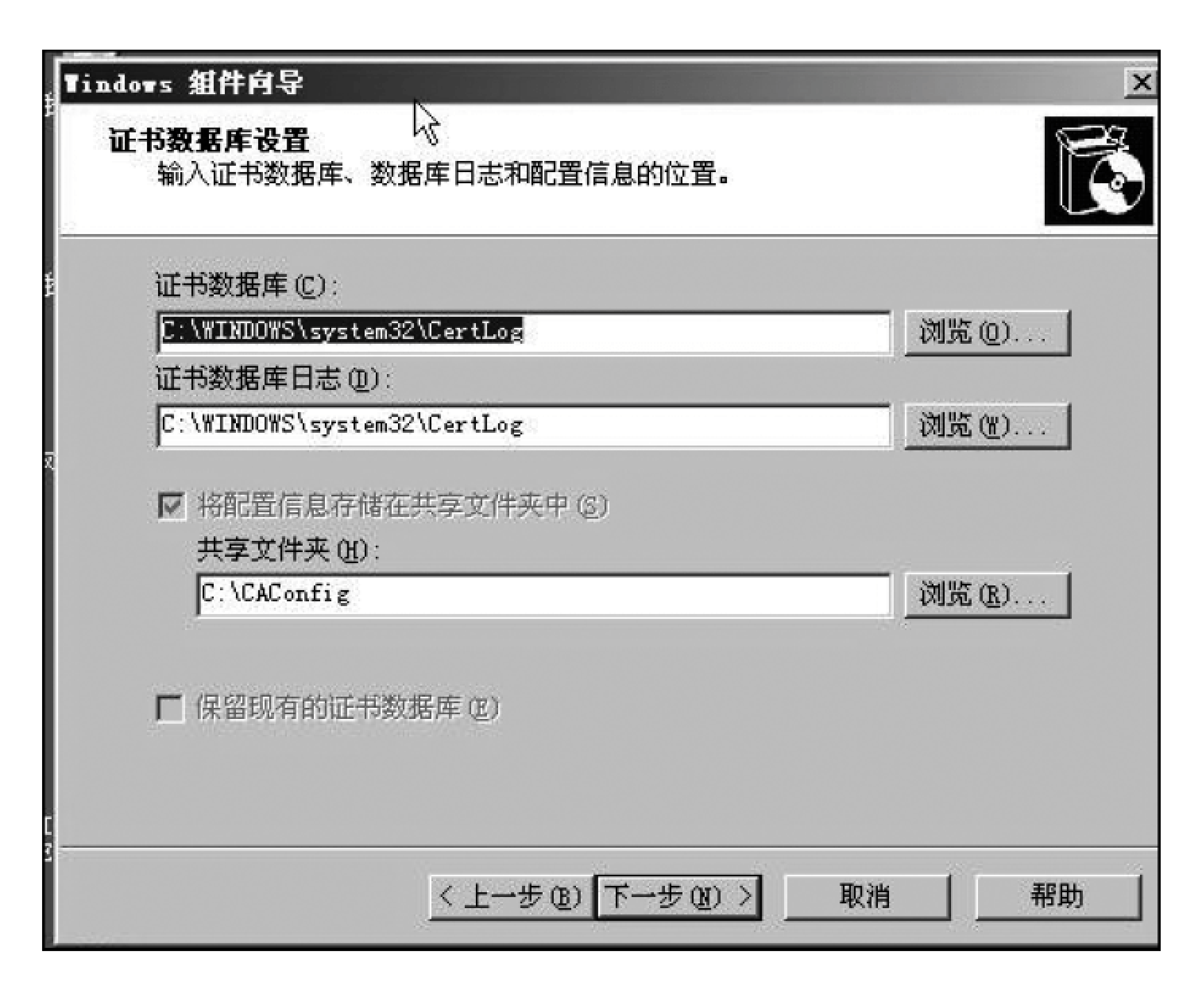


图 3-11　选择证书数据库及其日志信息的位置



图 3-12　安装证书时需要停止 Internet 信息服务



图 3-13 启用 ASP 支持以完成安装

(5) 打开证书颁发机构查看，如图 3-14 所示。



图 3-14 证书颁发机构

(6) 使用 IIS 查看默认站点关于 CA 证书的列表，如图 3-15 所示。



图 3-15 CA 证书列表

(7) 在 Web 服务器上设置 SSL,生成证书申请,如图 3-16 所示。

图 3-16 服务器配置 SSL

(8) 单击“安全通信”选项区域中的“服务器证书”按钮安装证书,选择新证书,如图 3-17 所示。



图 3-17 服务器证书申请

证书向导过程不一一赘述,填补完信息即可。

首先提交证书申请。

(1) 使用 IE 浏览器打开本地 Web 站点进入证书申请页面,如图 3-18 和图 3-19 所示。

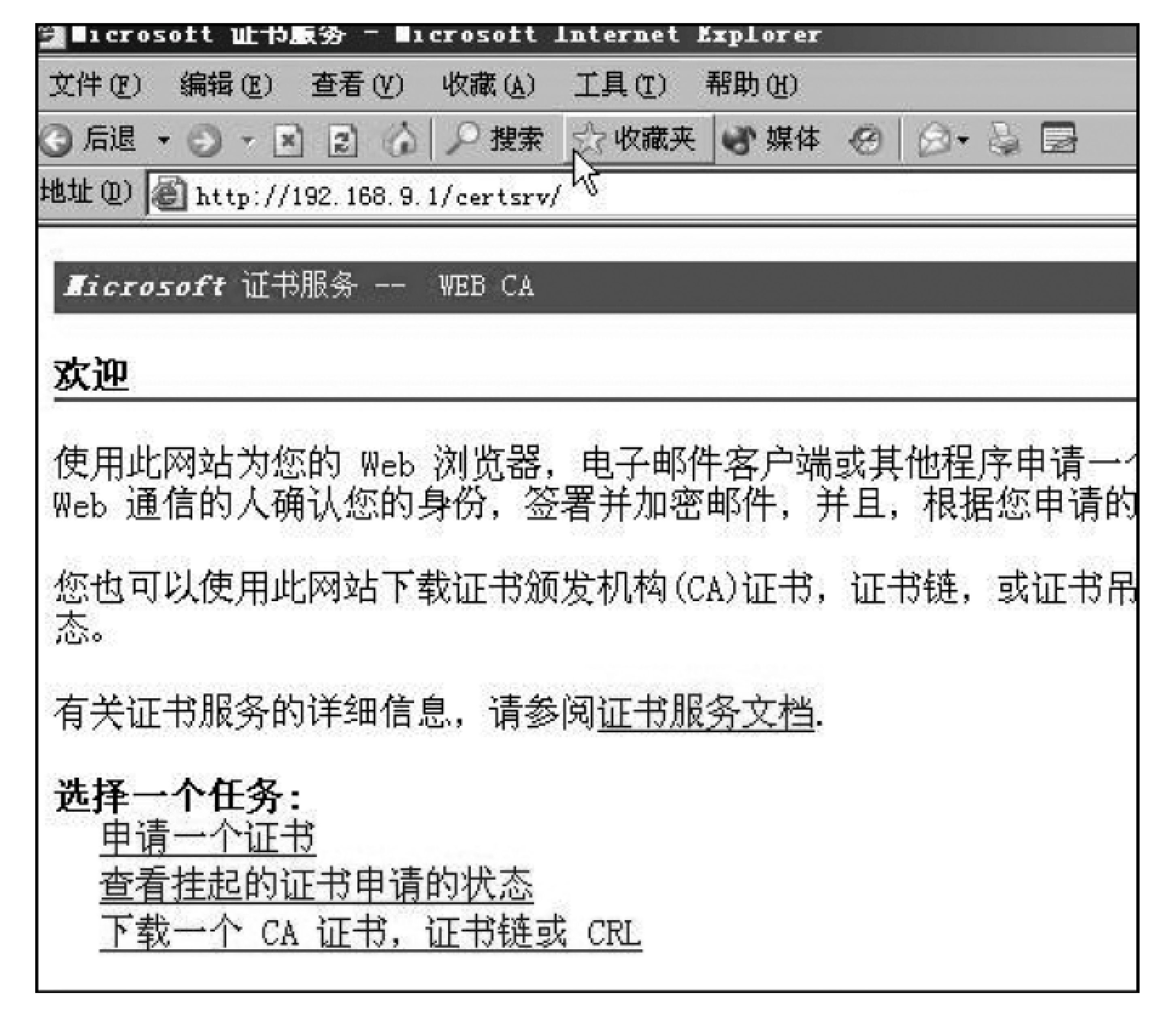


图 3-18 选择申请证书

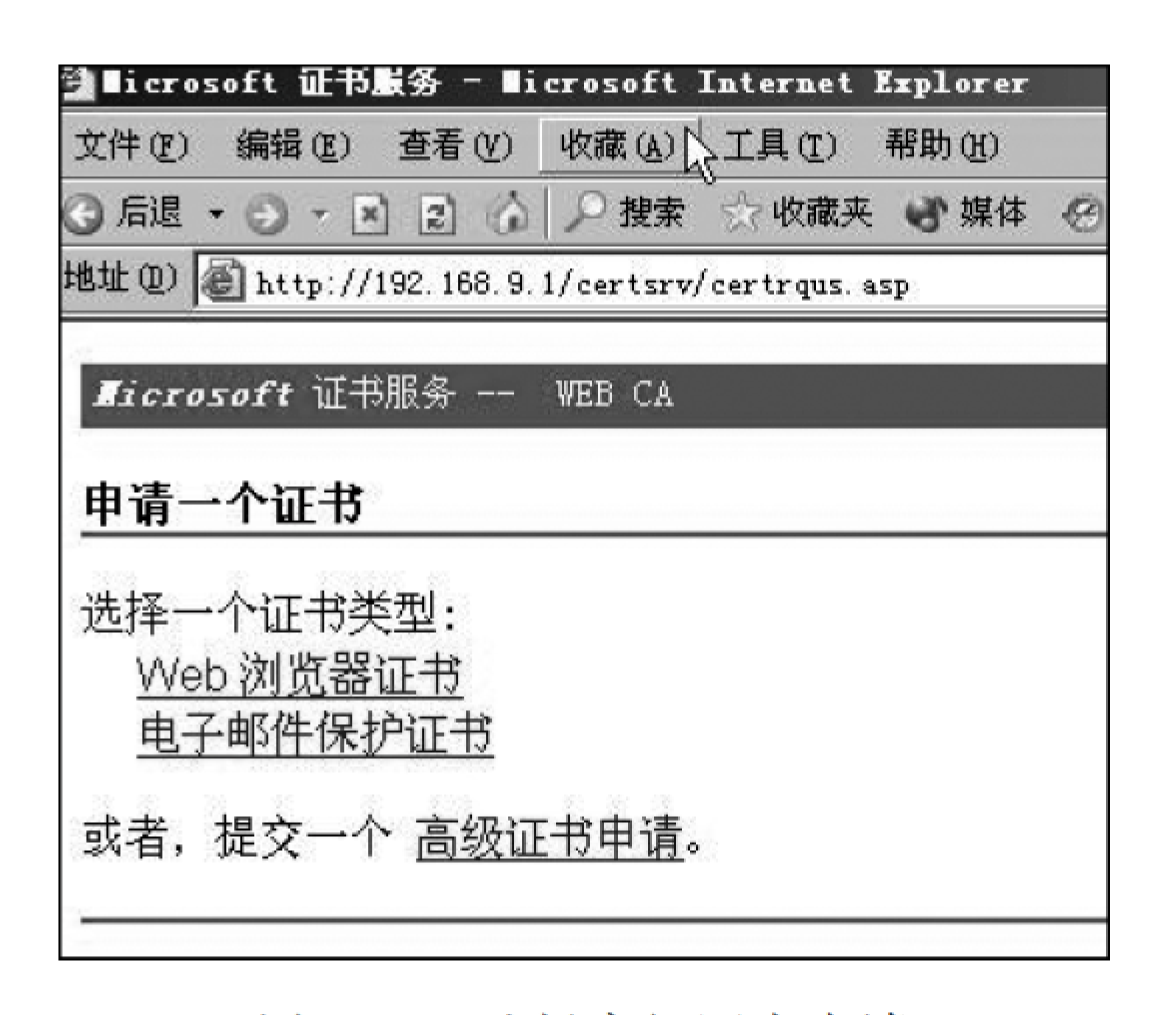


图 3-19 选择高级证书申请

（2）在提交一个证书申请中选择后者“BASE64 编码的证书申请”，如图 3-20 所示。

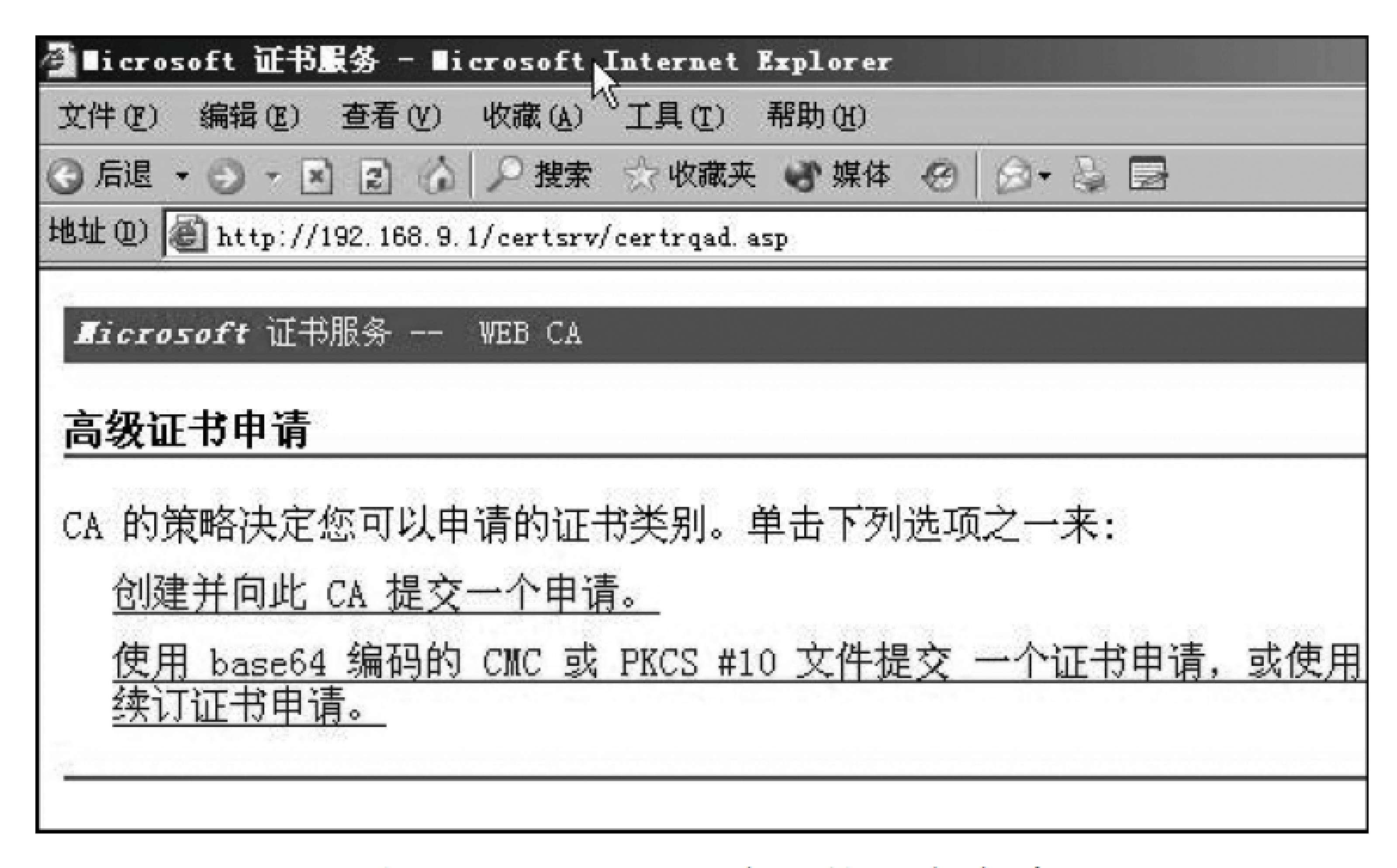


图 3-20 BASE64 编码的证书申请

(3) 此时弹出文本框,需要输入内容,如图 3-21 所示。

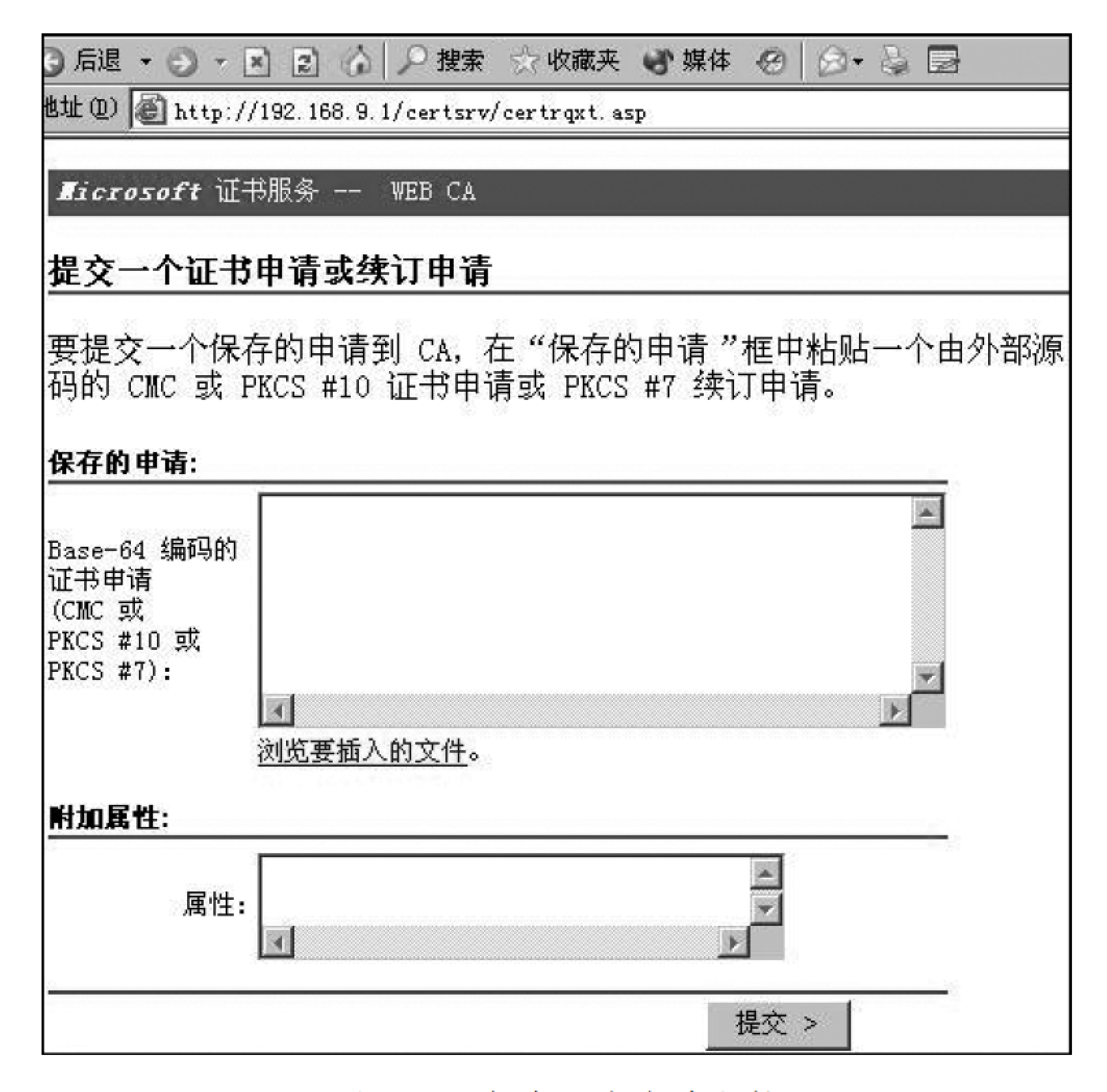


图 3-21 保存证书申请文件

(4) 找到之前保存的 certreq.txt,打开并把其内容全部复制到文本框中,然后单击“提交”按钮,完成以上步骤后证书被挂起,如图 3-22 所示。

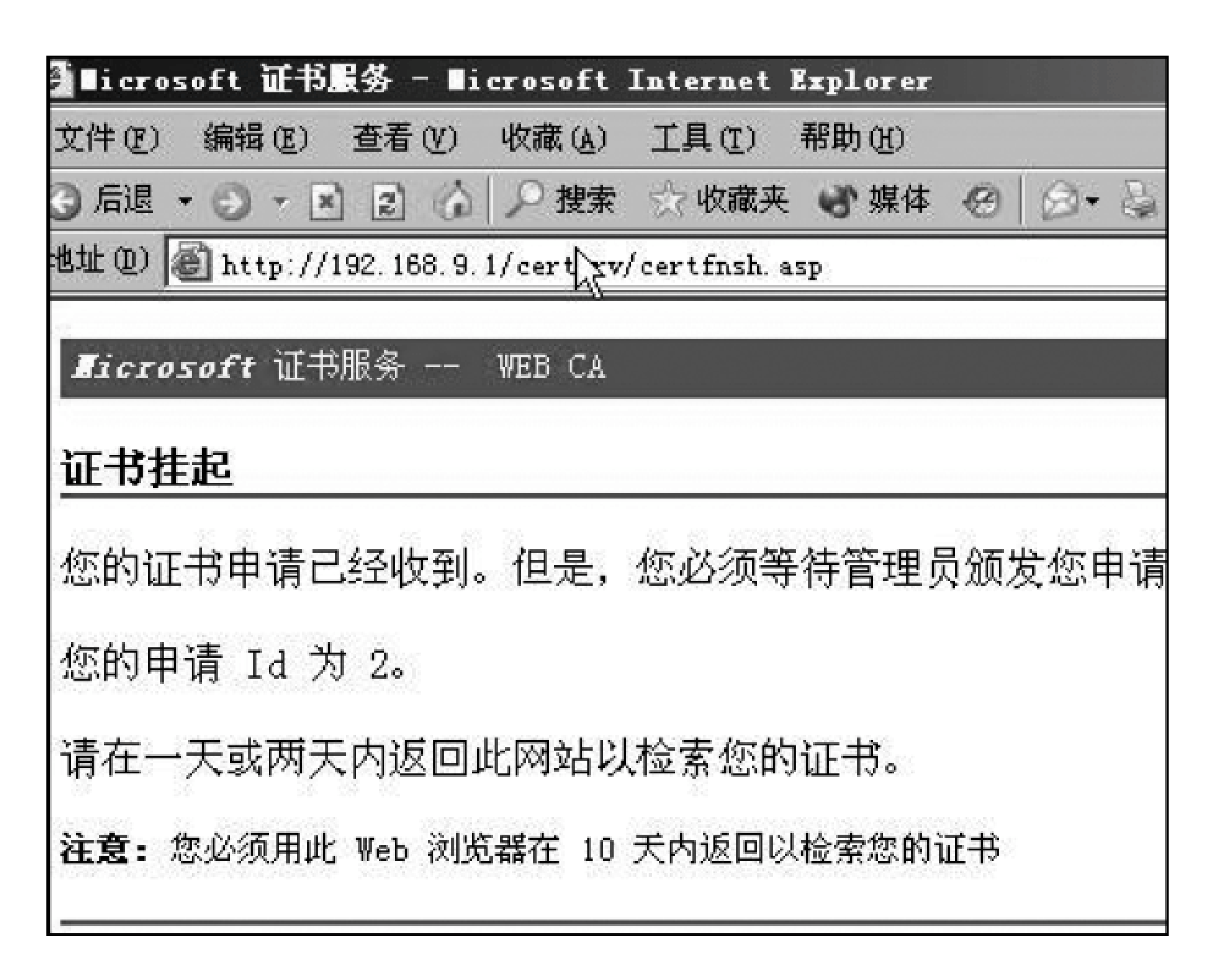


图 3-22 证书挂起

当证书申请被提交到证书颁发机构,下一步就要颁发证书了。

(1) 打开证书颁发机构工具,选择“挂起的申请”选项,如图 3-23 所示,可以看到刚才申请后被挂起的证书,然后选中挂起的证书,右击选择“颁发”命令。

(2) 使用 Web 形式访问证书申请页面,并选择下载证书,如图 3-24 所示。



图 3-23　颁发证书

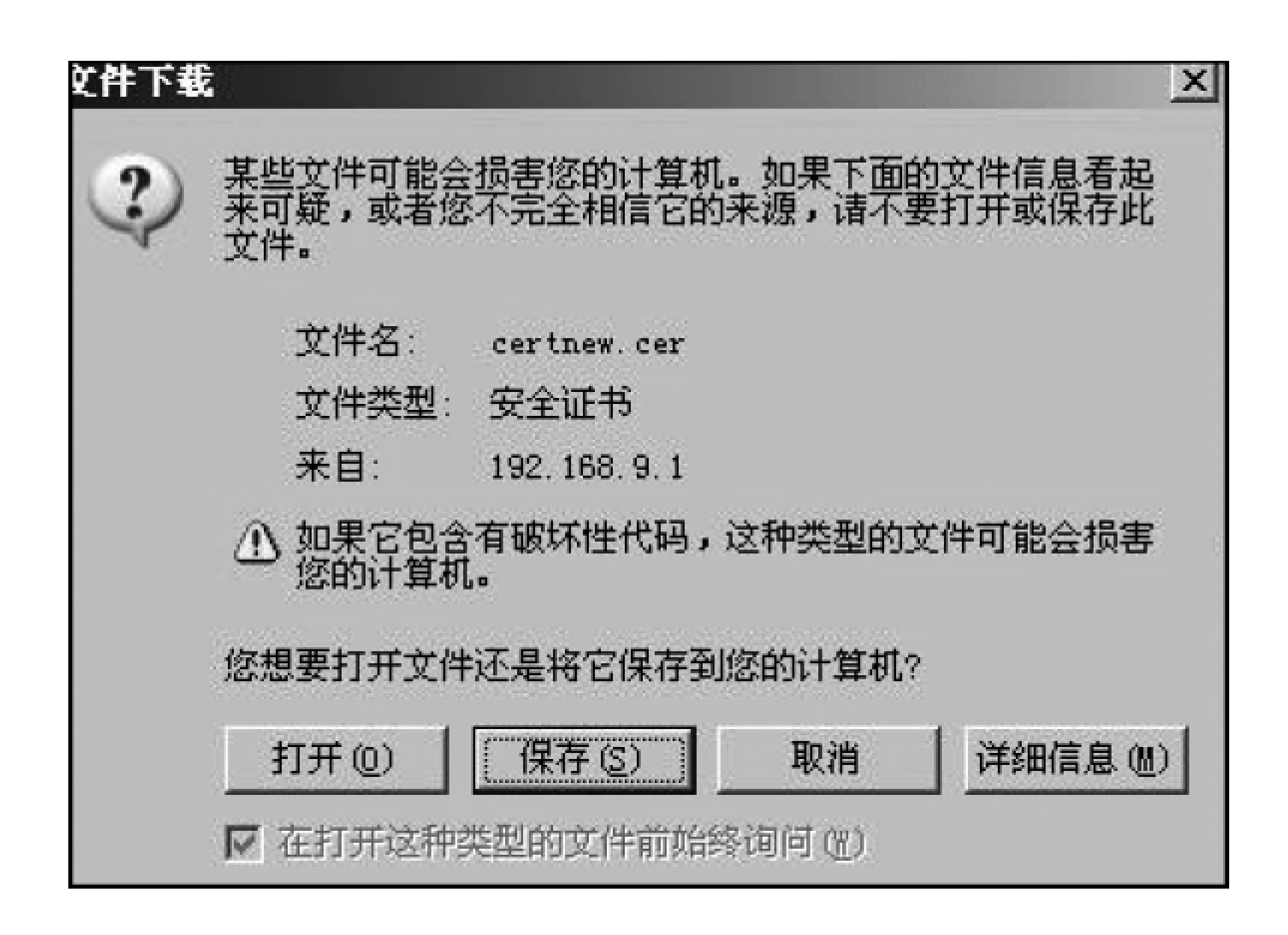


图 3-24　下载证书

(3) 在 Web 服务器上安装证书。

再次打开 IIS 服务器找到站点属性，再次单击"服务器证书"按钮，之后选择处理被挂起的请求，如图 3-25 所示。

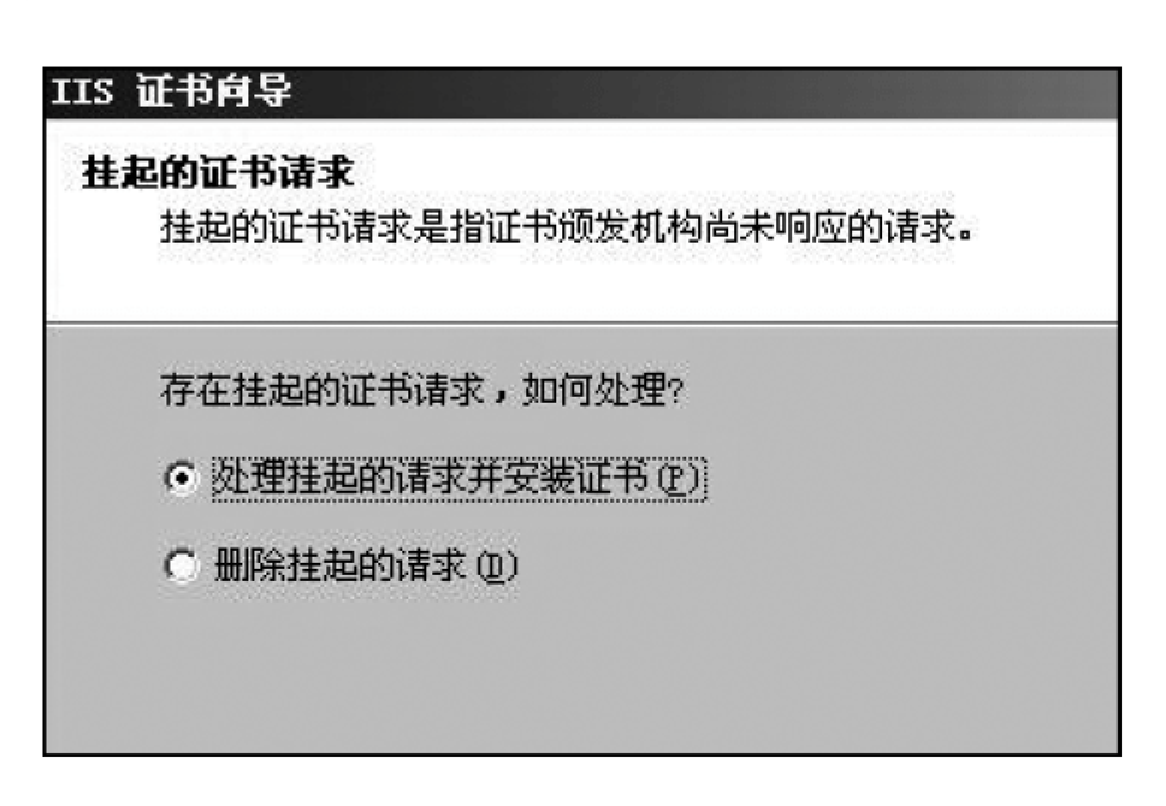


图 3-25　在 Web 服务器上安装证书

输入包含 CA 响应的文件的路径和文件名。

(4) 启用安全通道(SSL)。

打开站点属性，选择"安全通信"→"编辑"按钮，然后选中"要求安全通道"复选框即可，如图 3-26 所示。

(5) 使用 HTTPS 方式访问站点，如图 3-27 所示。

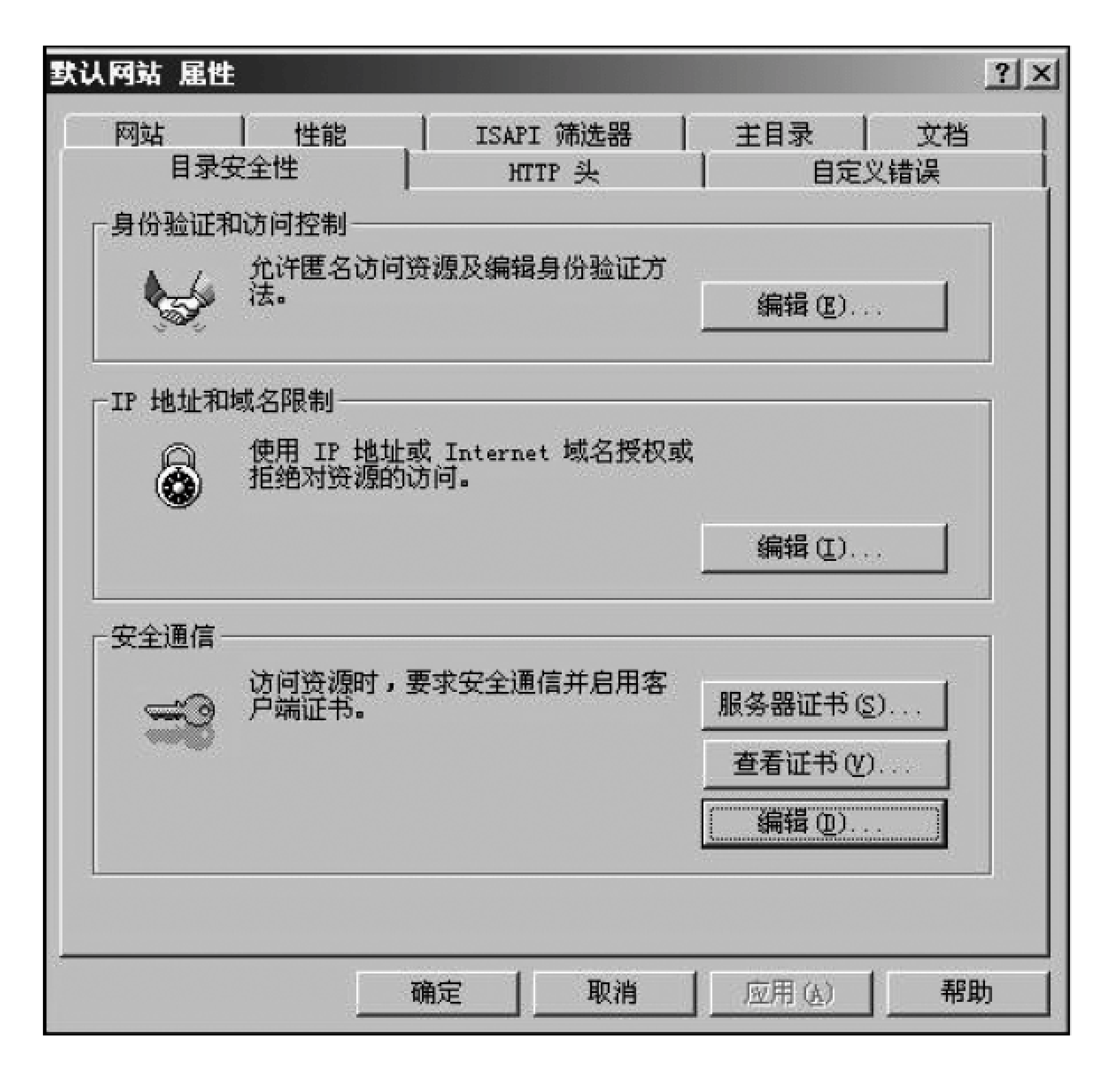




图 3-26　启用安全通道(SSL)

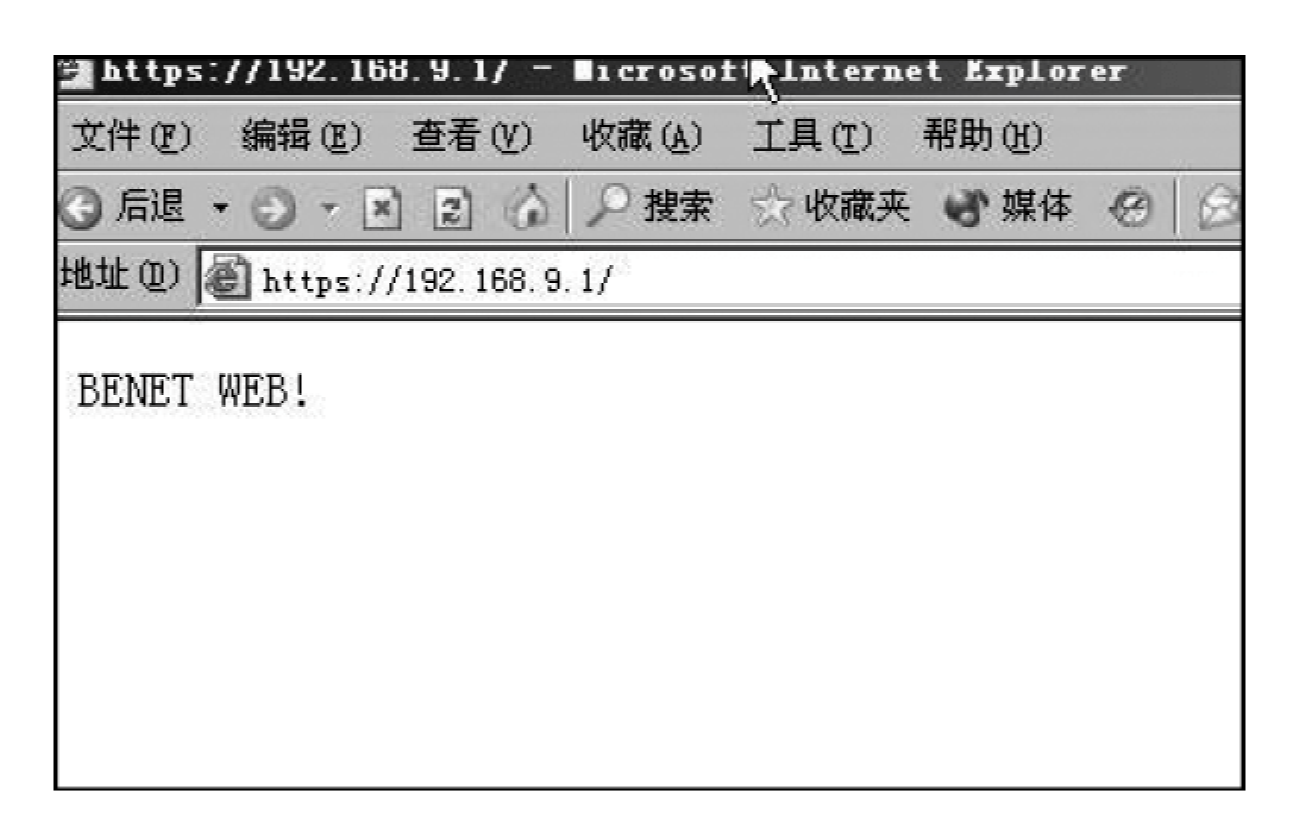


图 3-27　HTTPS 访问站点

【实验报告】

(1) SSL 协议的原理及运行机制；

(2) 基于 Web 的 SSL 连接的配置；

(3) 基于证书的身份证和 CA 服务器的搭建；

(4) WireShark 工具抓包分析 SSL 记录，并绘制 SSL 握手的流程。

3.3　小结

本章进行了 SSL 相关实验，对 SSL 协议的原理和结构有一定的认识和了解。通过 SSL 的配置实验，熟练掌握了对 Web 数据进行 SSL 安全传输的技术；同时熟悉了 X.509 v3 证书的格式以及申请证书的过程。

参 考 文 献

[1] 王志海等. OpenSSL 与网络信息安全：基础、结构和指令[M]. 北京：清华大学出版社，2007.
[2] 马小婷. 深入浅出密码学：常用加密技术原理与应用[M]. 北京：清华大学出版社，2012.
[3] 孙建国等. 网络安全实验教程[M]. 北京：清华大学出版社，2011.
[4] J. Frahim and Q. Huang. SSL Remote AccessVPNs [M]. 北京：人民邮电出版社，2009.
[5] Viega，John，Messier 等. Network Security with Open SSL [M]. O'Reilly Media，2002.
[6] Oppliger，Rolf. SSL and TLS：Theory and Practice [M]. Artech House Publishers，2009.
[7] Rescorla，Eric. SSL and TLS：Designing and Building Secure Systems [M]. Addison-Wesley Professional，2000.
[8] 周敬利，曾海鹏. SSL VPN 服务器关键技术研究[J]. 计算机工程与科学，2005(6).

第 4 章 缓冲区溢出攻击初级实验

在现有的攻击类型中，缓冲区溢出攻击是所有攻击类型中最为常见的一种。在所有的漏洞中，缓冲区溢出漏洞占了远程网络攻击中的绝大多数。缓冲区溢出攻击之所以成为一种常见的攻击手段，其原因在于缓冲区溢出漏洞容易实现并且最为普遍，另外，攻击者通过在远程靶机中植入并执行攻击代码，从而获取被攻击主机的权限，并以此做自己想做的任何事情。

目前的缓冲区攻击方式有很多种，其中最常见的有栈溢出、堆溢出、整型溢出、格式化字符串溢出及文件流溢出等。这些溢出攻击对原有程序具有很大的危害，其中包括应用程序异常，系统不稳定甚至崩溃，甚至程序跳转到恶意代码，控制权被窃取。本实验通过使用 OllyDbg 调试 OD 漏洞，理解缓冲区溢出攻击原理，了解栈溢出的攻击过程，从而在实际的写程序中尽量避免缓冲区溢出漏洞。

4.1 栈溢出原理

4.1.1 预备知识

栈是一段连续的内存，程序可以将数据压入栈中，也可以将数据从栈顶弹出，栈顶由称为 esp 的寄存器进行定位。压栈的操作使得栈顶的地址减小，弹出的操作使得栈顶的地址增大。其生长方向与内存的生长方向正好相反，从高地址向低地址生长，并且每一个线程有自己的栈。图 4-1 为内存分布示意图。

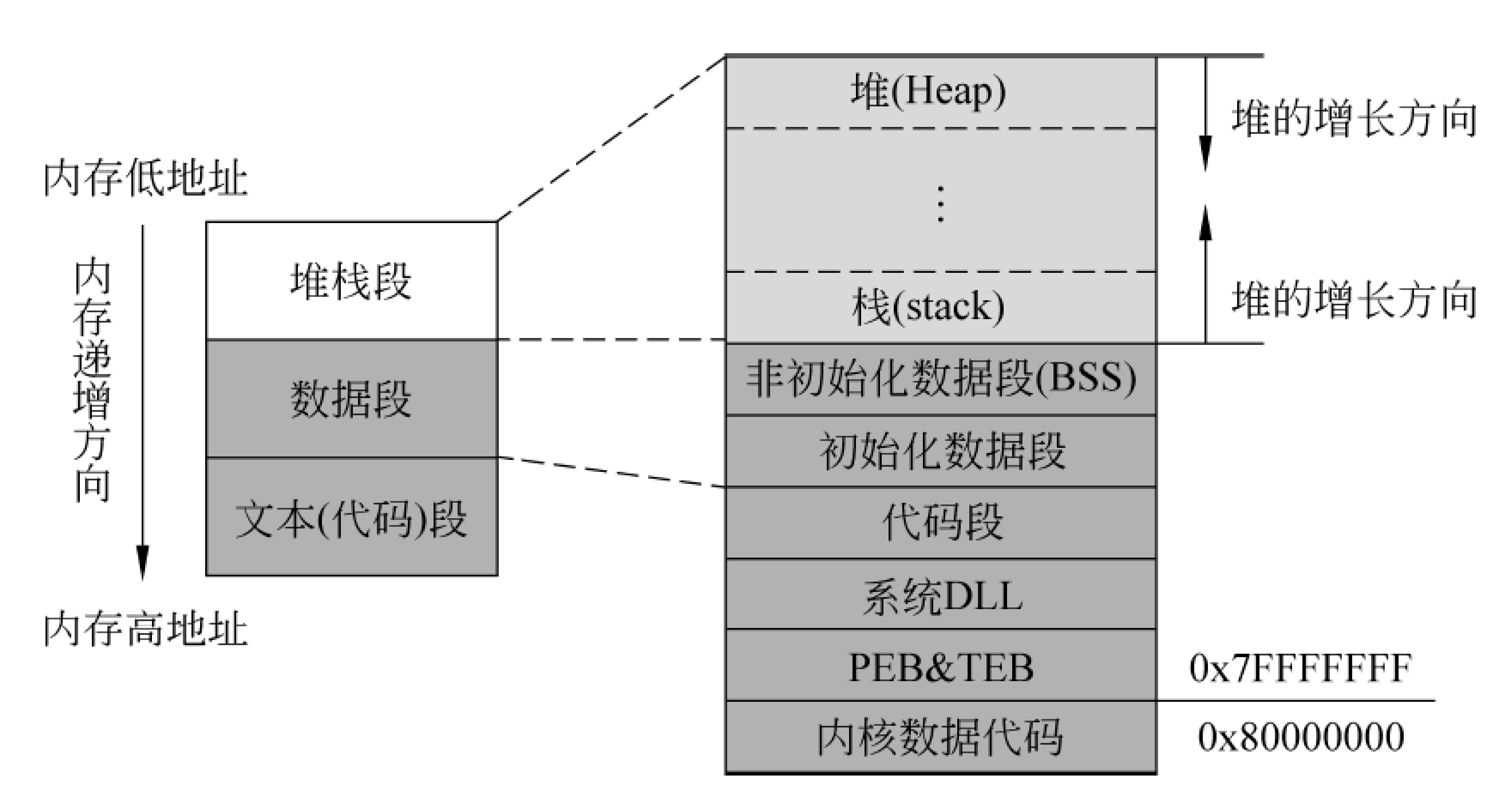


图 4-1 内存分布示意图

此处有3个重要的寄存器：堆栈指针寄存器(ESP)、基地址寄存器(EBP)和指令指针(EIP)。下面分别介绍这3个寄存器的作用。EIP(32位指令指针)是下一条指令在代码段(CS:指令或代码存放的位置)里的偏移,一般代码是不能直接访问EIP值的。在每条指令被执行之后EIP值会自动增加一个值以指向下一条要执行指令的地址。当要调用子程序时,系统就需要知道下一条指令的地址以及如何返回原始调用地址。调用指令通常规定了要往EIP所加的值,并把它压入堆栈。而调用函数中的返回指令会把堆栈值弹出给EIP以恢复调用后下一条指令的执行。同样堆栈指针ESP指向堆栈的最高端,EBP可指向堆栈的任意位置。换句话说,EBP是具有固定偏移的局部变量和参数堆栈信息的偏移参考寄存器;ESP地址经常发生变化,因为它一直指向堆栈的顶端。任何压栈和出栈操作均会影响ESP的值。

除此之外,必须理解函数的调用过程。通常一个函数调用过程有如下步骤：把参数压入栈,保存指令寄存器中的内容,作为返回地址,放入堆栈当前的基址寄存器,把当前的栈指针(ESP)复制到基址寄存器,作为新的基地址,为本地变量留出一定空间,把ESP减去适当的数值。

4.1.2 缓冲区溢出攻击原理

缓冲区溢出攻击发生的原因是因为缓冲区在栈中分配,然而复制的数据过长,因此覆盖了函数的返回地址或其他一些重要数据结构、函数指针,从而导致了溢出攻击的发生。

4.1.3 缓冲区溢出防御方法

避免缓冲区溢出攻击大致有以下几种方法：堆栈不可执行,寻址空间随机分布,对源代码进行安全漏洞分析,改善编译器,以及使用安全的编程语言和更安全的函数库。

1. 堆栈不可执行

由于现在大多数缓冲区溢出攻击基本上都是堆栈溢出攻击,其通过将攻击代码植入到堆栈中,然后在执行攻击代码。因此很容易想到一种最为简单的防御方法就是设置堆栈区不可执行。

2. 寻址空间随机分布

由于特定系统函数的入口地址是可以事先确定的。因此可以通过寻址空间随机分布来避免发生溢出攻击。在Windows在它的最新操作系统中采用了ASLR技术来防御此类缓冲区溢出,在操作系统启动时,随机从256个地址空间中选出一个载入DLL/EXE。这样攻击方就难以事先确定系统函数的入口地址。

3. 对源代码进行安全漏洞分析

利用专用安全分析工具软件,对源代码进行词法、语法上的分析,根据已知缓冲区溢出漏洞的数据库,找出程序中可能存在的安全隐患,并给予相应的解决提示,帮助编程人员迅速地修改程序。

4. 改善编译器

通过改进编译器,增加缓冲区完好检验、输入数据的边界检查等,防止缓冲区溢出,让

存在隐患的程序编译不能通过。这种方法为防止缓冲区溢出攻击提供了较好的解决方法，但是它需要程序的源代码，需要对其重新编译，增加了程序运行时的额外开销，存在一定的误报率，并且没有真正除去程序中的安全漏洞，当程序在普通环境运行时，漏洞依然存在。

5. 使用安全的编程语言

缓冲区溢出攻击发生的重要原因之一是使用了不安全的函数，例如 strcpy、scanf 等。程序员在调用这些函数时，如果没有进行仔细的检查，就很容易留下受到缓冲区溢出攻击的漏洞。因此使用更安全的函数库对标准函数库中有漏洞的函数进行封装，加入安全检查。安全工具以动态链接库的形式存在于系统中，对于用户是透明的。当程序调用有缺陷的函数时，它们会自动加载安全检查，通过检查后才真正去调用该函数。

4.2　实验　缓冲区溢出攻击实验

【实验目的】

通过对 OllyDbgICE 的使用，理解缓冲区溢出攻击发生的原理。

【实验内容】

根据实验步骤，一步步地实践和理解缓冲区溢出的原理。

【实验环境】

PC 一台。

【实验参考步骤】

(1) 编译如下简单程序例子，生成 DEBUG 版程序。

```
#include <stdio.h>
#include <windows.h>
#include <string.h>
int fun(char * cpybuf){
    char Buf[8];
    strcpy(Buf,cpybuf);
    return 0;
}
int main(){
    MessageBoxA(NULL,"This is a Test","BOF",MB_OK);
    char buff[]="1234567";
    fun(buff);
    return 0;
}
```

(2) 打开 OD，将上述生成的 DEBUG 程序 sample.exe 在 OD 中打开，图 4-2 是打开 sample.exe 的截图。

(3) 找到 main()函数的入口地址。对比代码，可以知道在地址 010F1488 处调用了

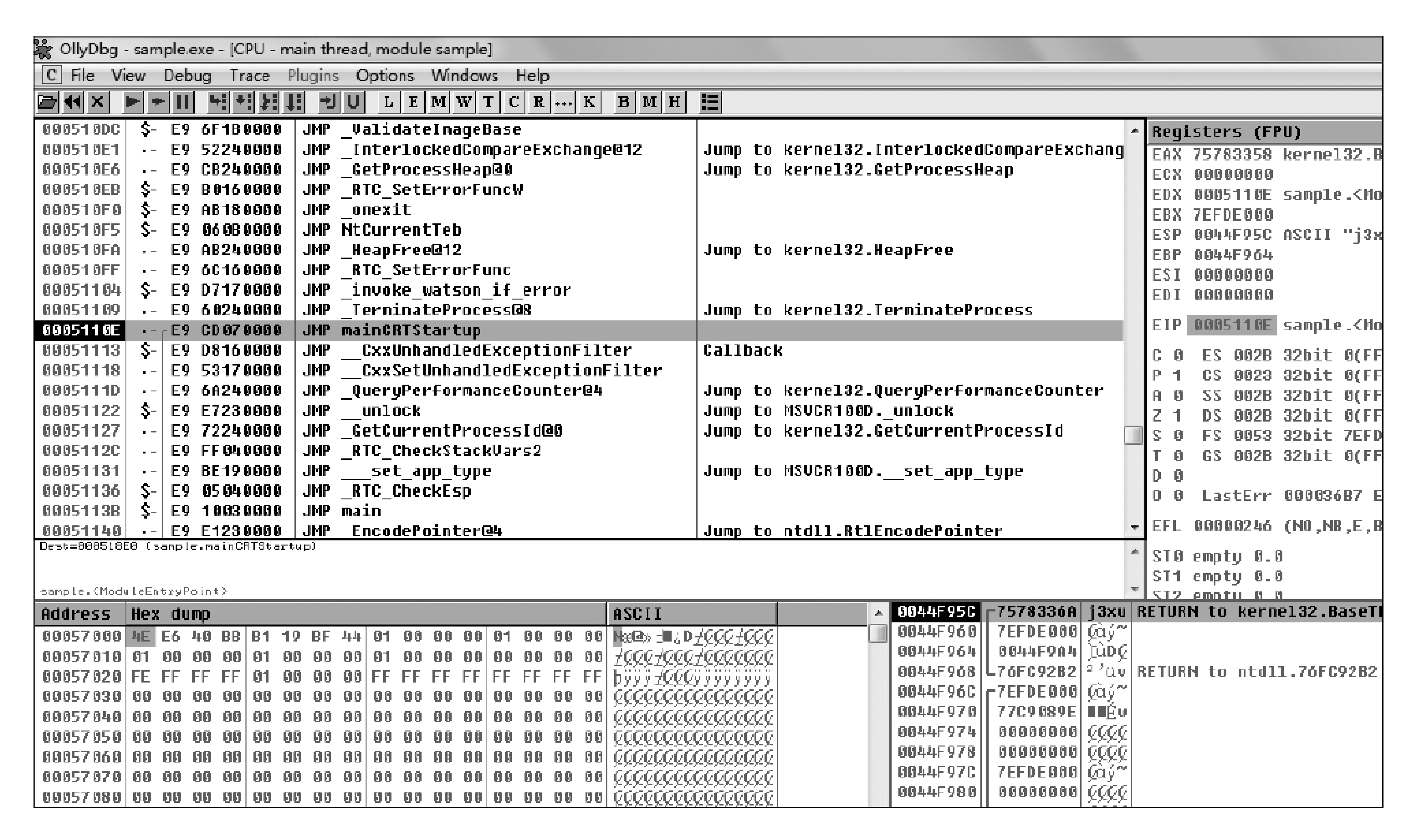


图 4-2　打开 sample.exe

动态链接库 User32.dll 中的 API 函数 MessageBox()，之后的连续的 MOV 指令则是初始化数组操作，并将初始化好的数组地址传递给 EAX，由它做参数传递，如图 4-3 所示。

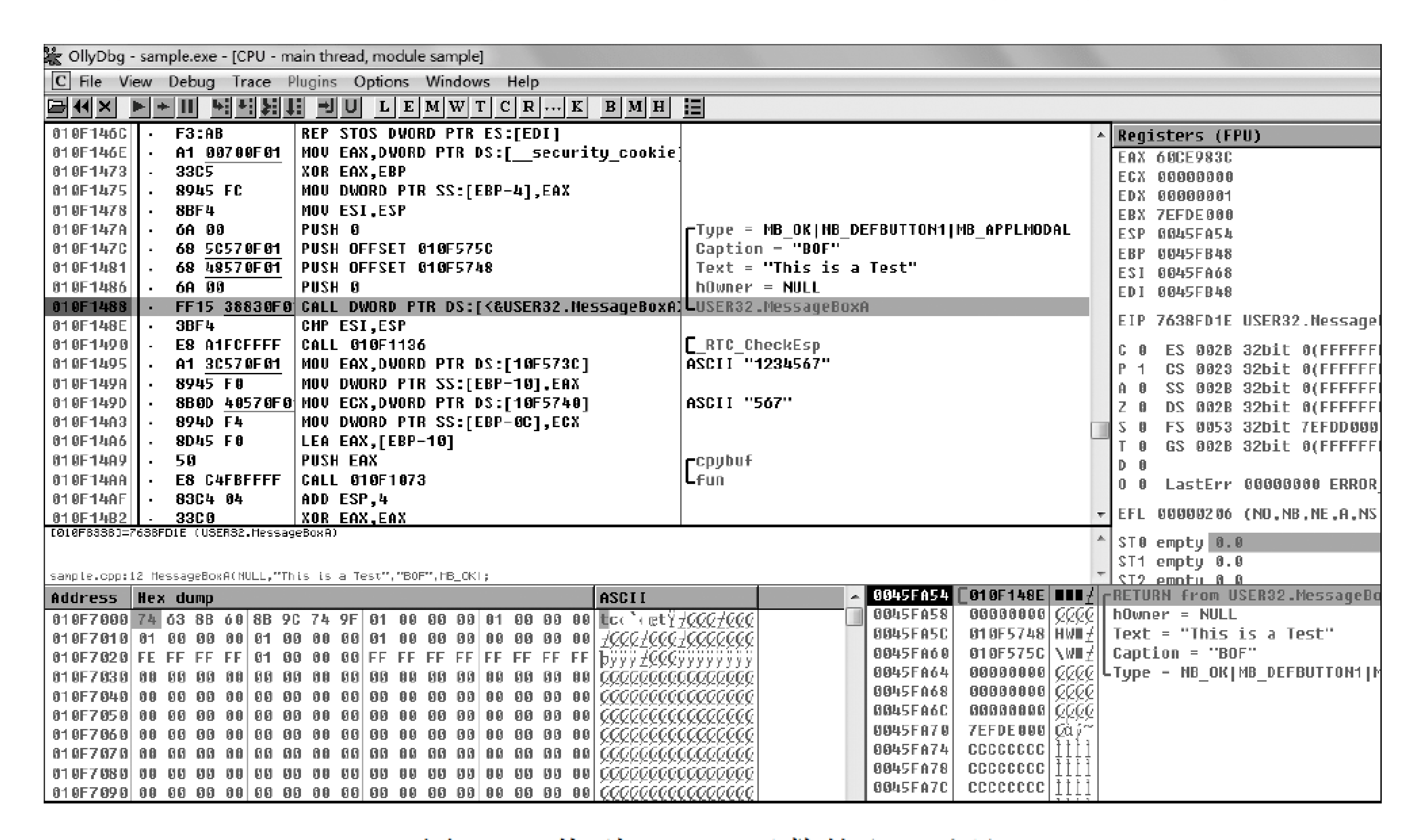


图 4-3　找到 main()函数的入口地址

(4) 单步跟踪至 010F13A0，即将进入 fun()函数，应注意堆栈的变化情况。F7 单步跟踪进入到 fun()函数内部，如图 4-4 所示。

显示了整个 fun()函数的反汇编代码情况。此刻注意堆栈的栈顶的情况，栈顶 0014F678 存放内容为 010F14AF，这正是图 4-3 中 main()函数中执行 fun()函数的下一句代码的地址，紧接着继续跟踪，看是否这个栈顶的内容是用于返回 main()函数。

(5) 单步跟踪到 fun()的最后一条指令 ret n 查看程序是如何回到 main()函数的，单步 F7 执行至 011714E2，指令为 RETN，查询汇编指令手册可知，是将栈顶值出栈给 EIP，如图 4-5 所示。

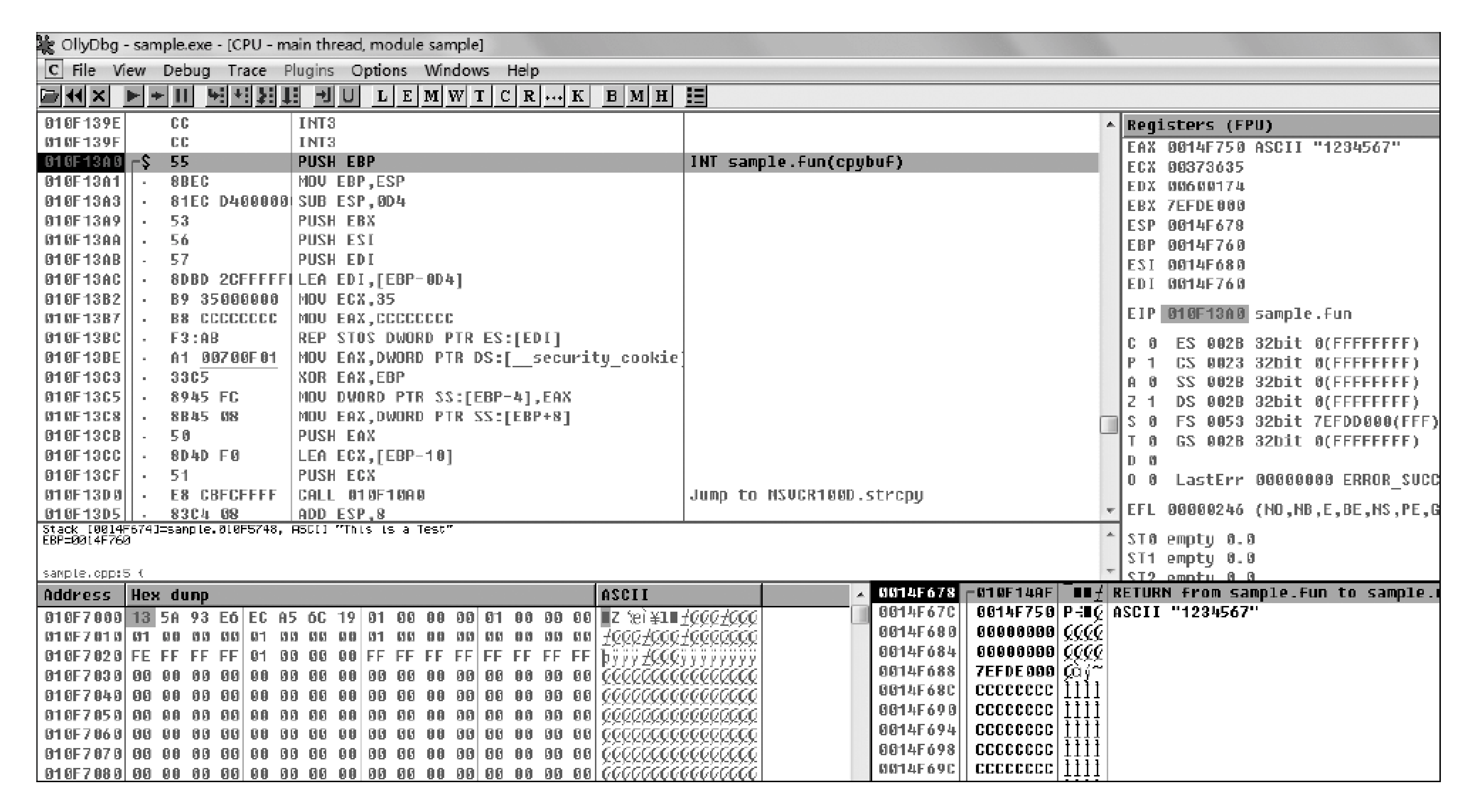


图 4-4　单步跟踪进入到 fun()函数内部

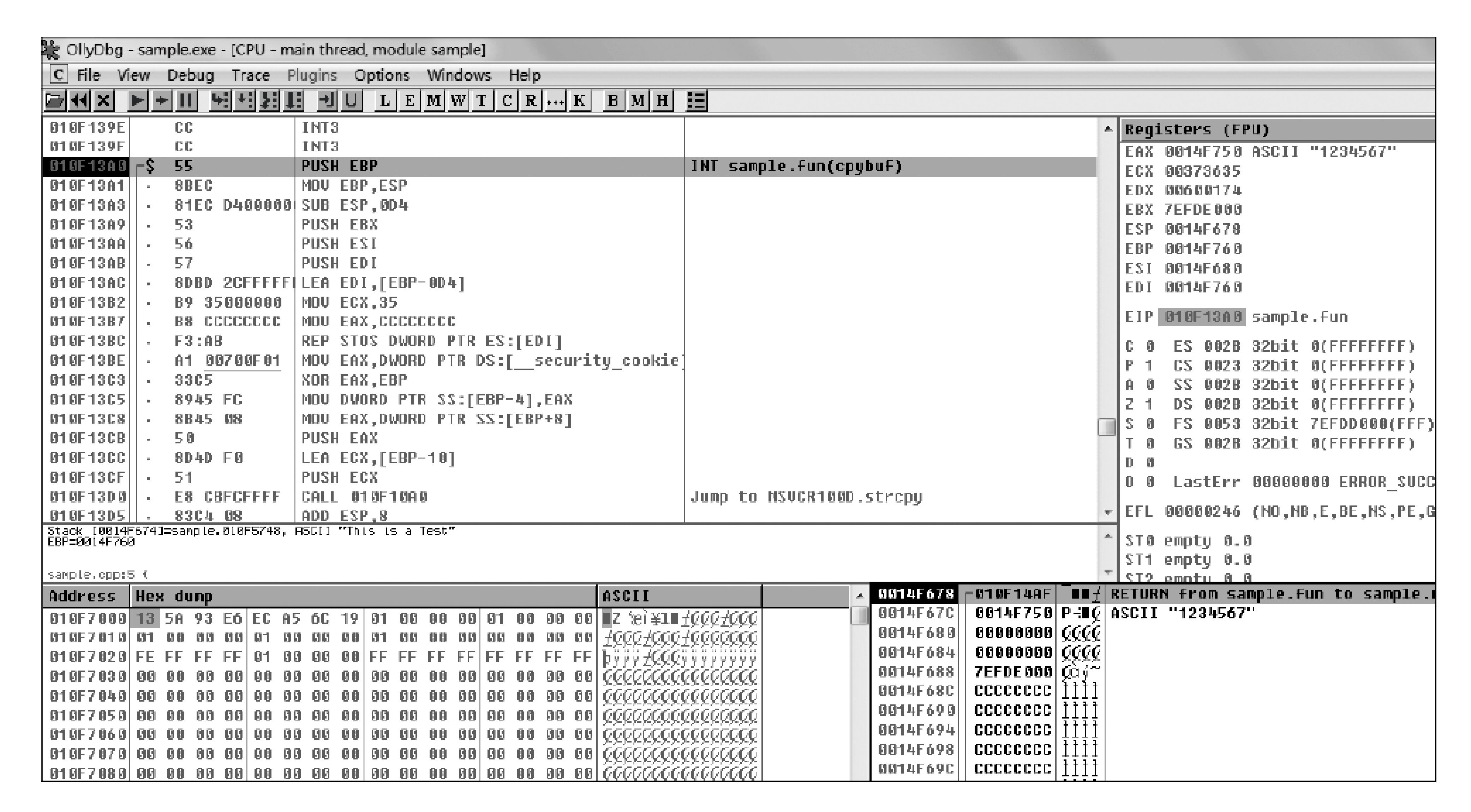


图 4-5　单步跟踪 ret n 指令

(6) 开始攻击演示。

修改源代码中的 buff[]初始化定义 char buff[]="1234567"为 char buff[]="1234567aaaaaaabbbbbbbbbbb",编译生成相应的 Debug 程序,然后放在 OD 里面根据上述步骤进行调试,注意栈顶的返回地址,如图 4-6 所示。

这是改变数组内容后的 fun()的反汇编代码,注意到栈顶 0013FF18 存放 004010D9,这是 main()中执行完 fun()函数后执行的代码。

(7) strcpy 执行。

单步执行到 Call 00401100,这里执行的是函数 strcpy(),查看执行 strcpy 后的结果,如图 4-7 所示。

此时可以发现 EBP 的值为 0014F7A8(ASCII 码为 bbbbbbbb)。

(8) 攻击发生。

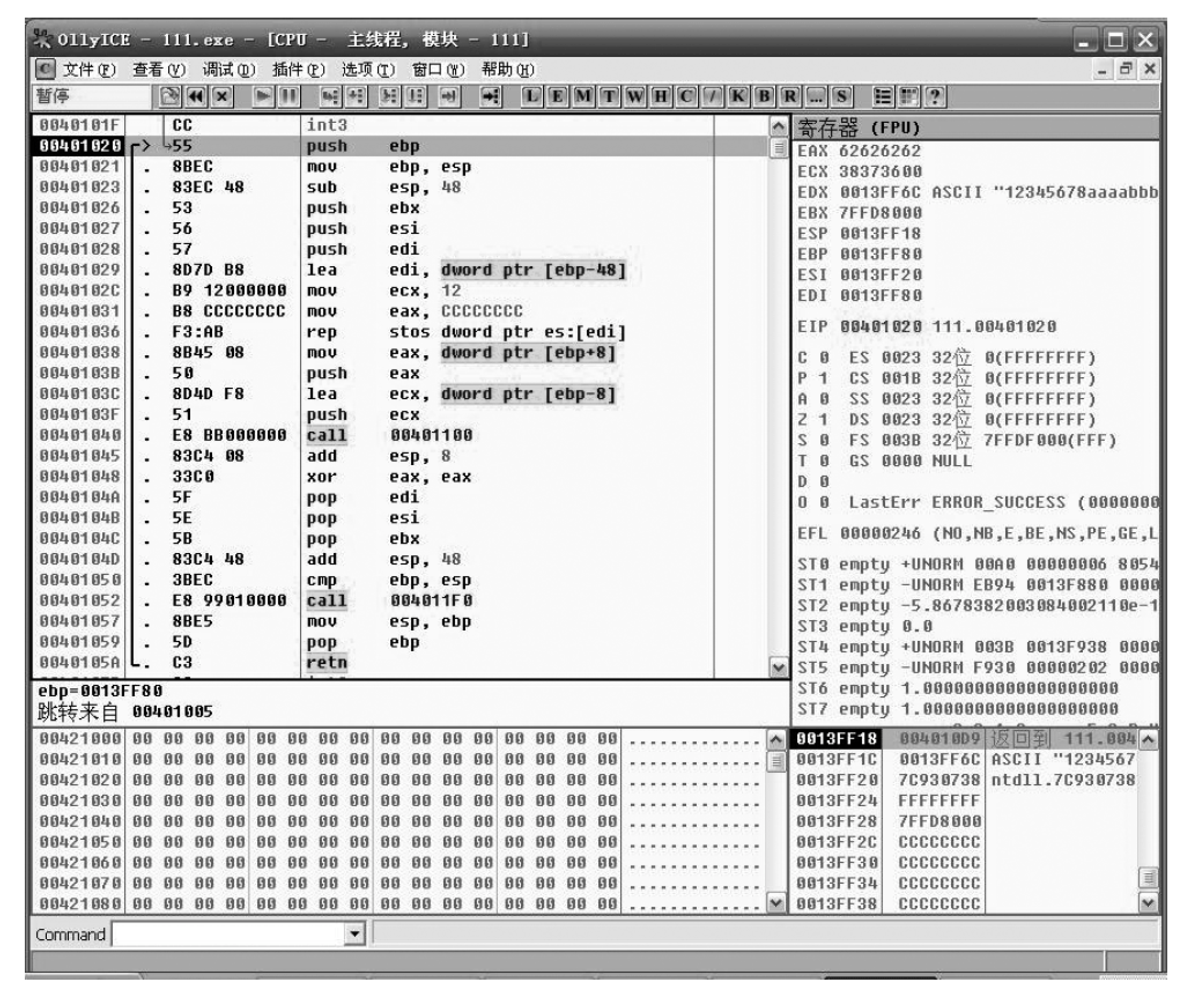


图 4-6　开始攻击

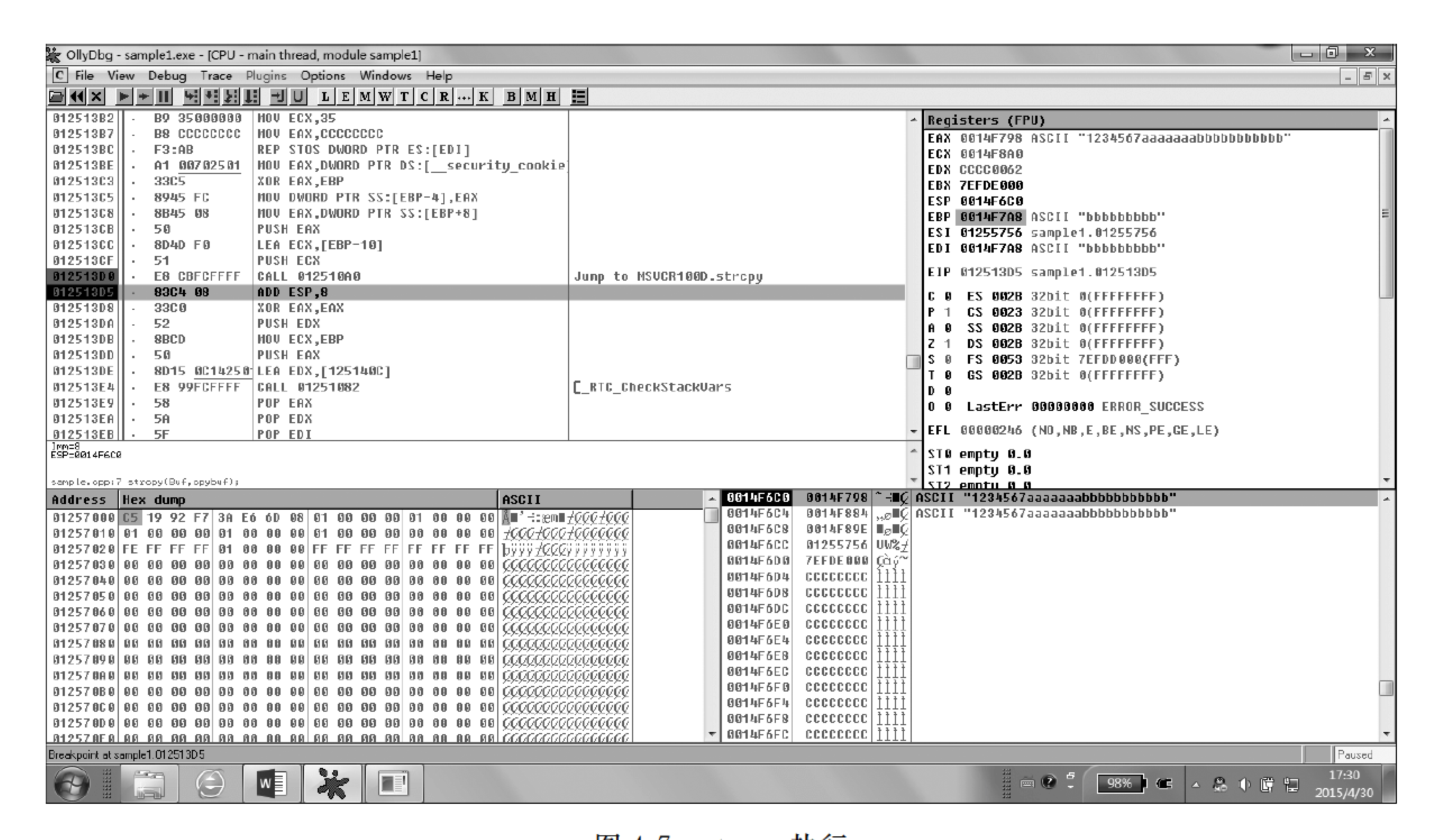


图 4-7　strcpy 执行

继续单步执行到 ret n 指令，栈顶正是 62626262，如果再执行 EIP->62626262，由于 62626262 不存在内容，导致程序执行出错，如图 4-8 所示，攻击发生。

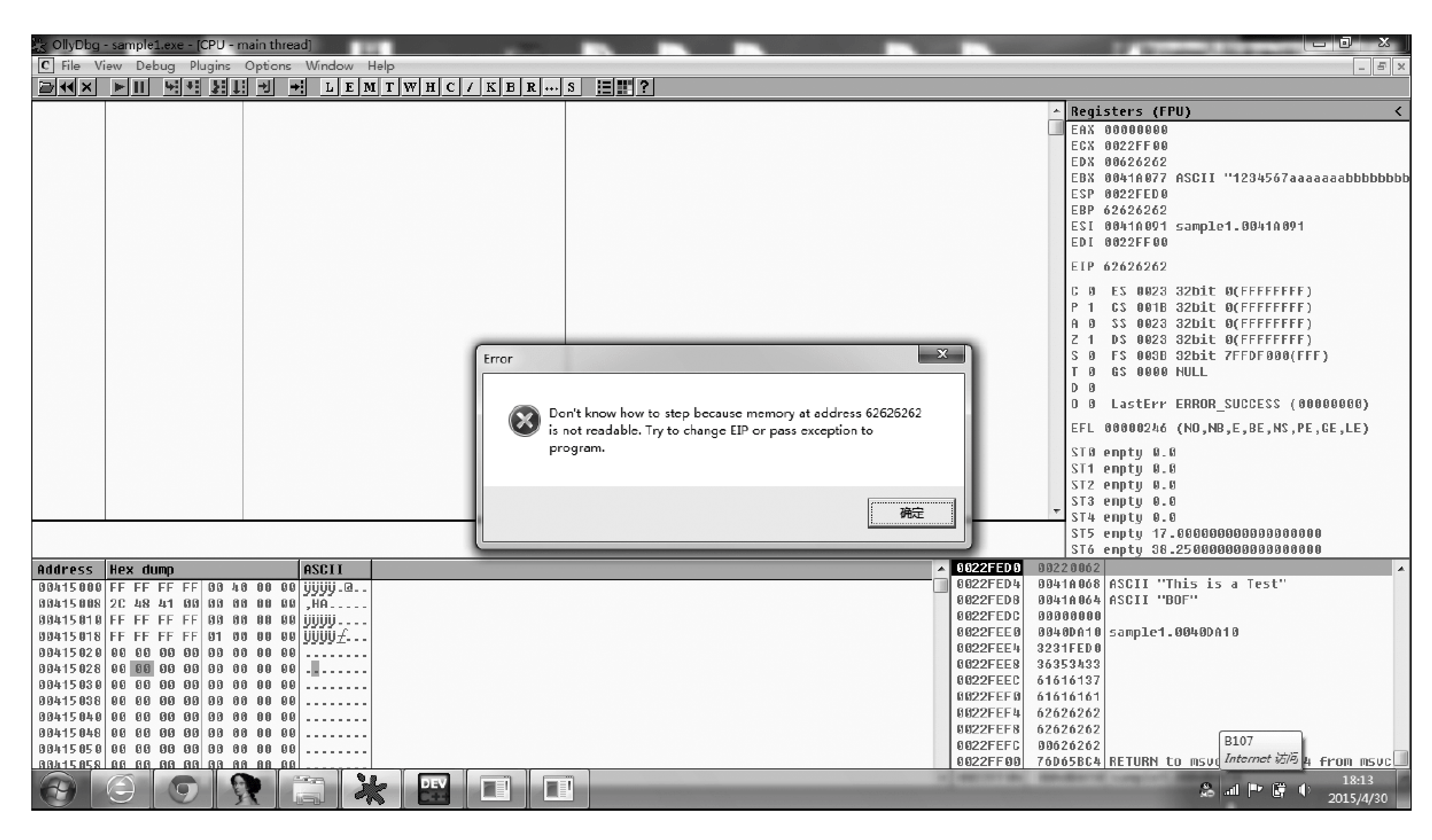


图 4-8　攻击发生

【实验报告】

(1) 缓冲区溢出攻击的原理；

(2) OllyDbg 调试器的使用步骤；

(3) 对简单的程序实施缓冲区溢出攻击。

4.3　小结

缓冲溢出攻击是一种经典的也是最具挑战性的攻击方式，是用于渗透对方主机的有力武器。通过本章实验，能对缓冲溢出有更深的认识，这对无论是代码开发还是安全相关工作都会起到很大的帮助作用。

参 考 文 献

[1]　孙建国等. 网络安全实验教程[M]. 北京：清华大学出版社，2011.

[2]　J. Frahim and Q. Huang. SSL Remote Access VPNs [M]. 北京：人民邮电出版社，2009.

[3]　蔡勉. 缓冲区溢出攻击：检测、剖析与预防[M]. 北京：清华大学出版社，2006.

[4]　黑客防线：缓冲区溢出攻击与防范专辑[M]. 北京：人民邮电出版社，2009.

[5]　P. Engebretson. 渗透测试实践指南：必知必会的工具与方法(原书第 2 版)[M]. 姚军，姚明，译. 北京：机械工业出版社，2014.

[6]　M. Shema. Web 应用漏洞侦测与防御[M]. 齐宁，庞建民等，译. 北京：机械工业出版社，2014.

[7]　福斯特等. 缓冲区溢出攻击：检测、剖析与预防[M]. 蔡勉，译. 北京：清华大学出版社，2006.

[8]　吴世忠，郭涛，董国伟等. 软件漏洞分析技术[M]. 北京：科学出版社，2014.

第 5 章 Radius 综合实验

远程认证拨号用户服务(Remote Authentication Dial-In User Service,RADIUS)是一种分布式的、客户端/服务器结构的信息交互协议,能保护网络不受未授权访问的干扰,常被应用在既要求较高安全性,又要求维持远程用户访问的各种网络环境中。

该综合实验是以 VMware 虚拟机为平台,Windows Server 2003 为服务器操作系统,最后在这一服务器上搭建以 Radius 协议的应用为核心,附带在其搭载的服务器上开通一系列的服务(如 DNS 域名服务、DHCP 服务以及 VPN 服务等)而形成的综合服务器。这样形成一个综合协议服务的服务器平台,达到了多协议多服务可以融合的目的。同时,对书本上的多种协议进行了实践性增强的应用。

同时综合实验又在多协议服务融合的基础上,增加了虚拟机搭建部分以及 Windows Server 2003 网络配置与管理的相关部分,充分为网络协议以及计算机网络等课程提供了理论联系实践的机会。该综合实验为网络以及协议学习者创造了一个内容丰富、融合性强、知识面广、联系紧密的锻炼机会。

整个综合实验分为三个子实验:

(1) Windows Server 2003 的配置以及 DNS、DHCP、HTTP 和 FTP 等服务器的配置;

(2) Radius 服务器与 VPN 服务器的搭建;

(3) 编写端口扫描小程序,对搭建的服务器进行服务端口开放自检测。

具体工作流程详见图 5-1。

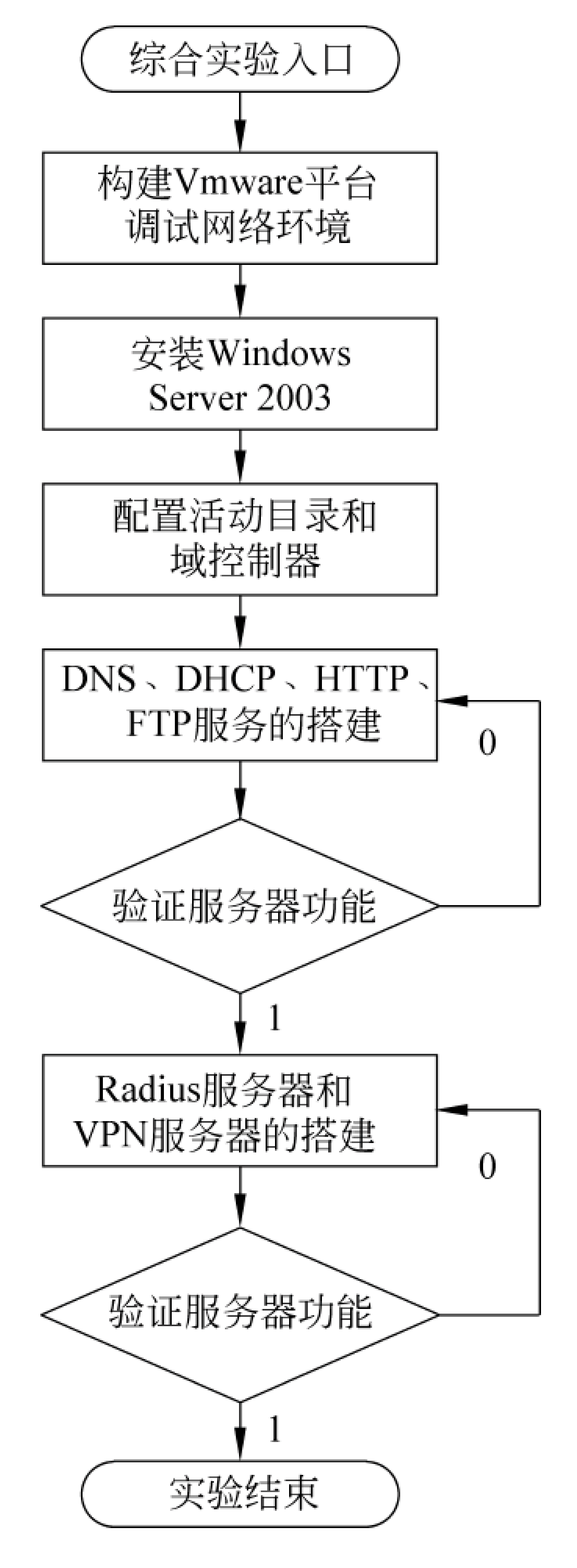


图 5-1 综合实验系统结构流程图

5.1 实验目的

本实验的核心目的,是学会如何在 Windows Server 2003 的操作系统平台上进行安全配置以及服务器的搭建。在进行配置之前,首先要对 Radius 协议的原理、VPN 虚拟专网通信原理、DNS、DHCP、HTTP、FTP 等服务的工作原理有所了解,进一步巩固理论知识的掌握的同时,深化知识结构体系。而如何配置才能安全有效地实现 Radius 的远程客

户端接入，是本实验最终的目的。

5.2　实验内容

Radius 是目前应用最广泛的 AAA 协议（认证 Authentication、授权 Authorization、计账 Accounting，一个提供网络安全的系统），应用包括普通电话上网、ADSL 上网、小区宽带上网、IP 电话、VPDN（虚拟专用拨号网）、移动电话付费等业务。

本实验的任务就是要在虚拟机上对 Windows Server 2003 进行安全配置，通过搭建 Radius 服务器和 VPN 服务器，以实现远程客户端的接入。具体任务分为三个子实验：

（1）为 Windows Server 2003 的安全进行系统的配置，其中包括两部分：

- 活动目录和域控制器的安全配置；
- DNS、DHCP、HTTP 和 FTP 等服务的安全配置。

（2）Radius 服务器和 VPN 服务器的搭建。

（3）编写端口扫描小程序，对搭建的服务器进行服务端口开放自检测。

5.3　Windows Server 2003 的安全配置

5.3.1　活动目录及域控制器的配置

RADIUS 服务器将用户身份信息转送到一个认证服务器上，在域环境中，这个认证服务器是活动目录的域控制器。认证服务器响应 RADIUS 服务器的身份验证请求，然后 RADIUS 服务器响应 ISA 防火墙。

注意：在此例中，域控制器就是一个 RADIUS 服务器，可以把它放在域中的其他机器上。域控制器同时还作为 DHCP 服务器、DNS 服务器，之后，我们会使用这些服务。

接下来，如图 5-2 所示，先行完成对活动目录配置。

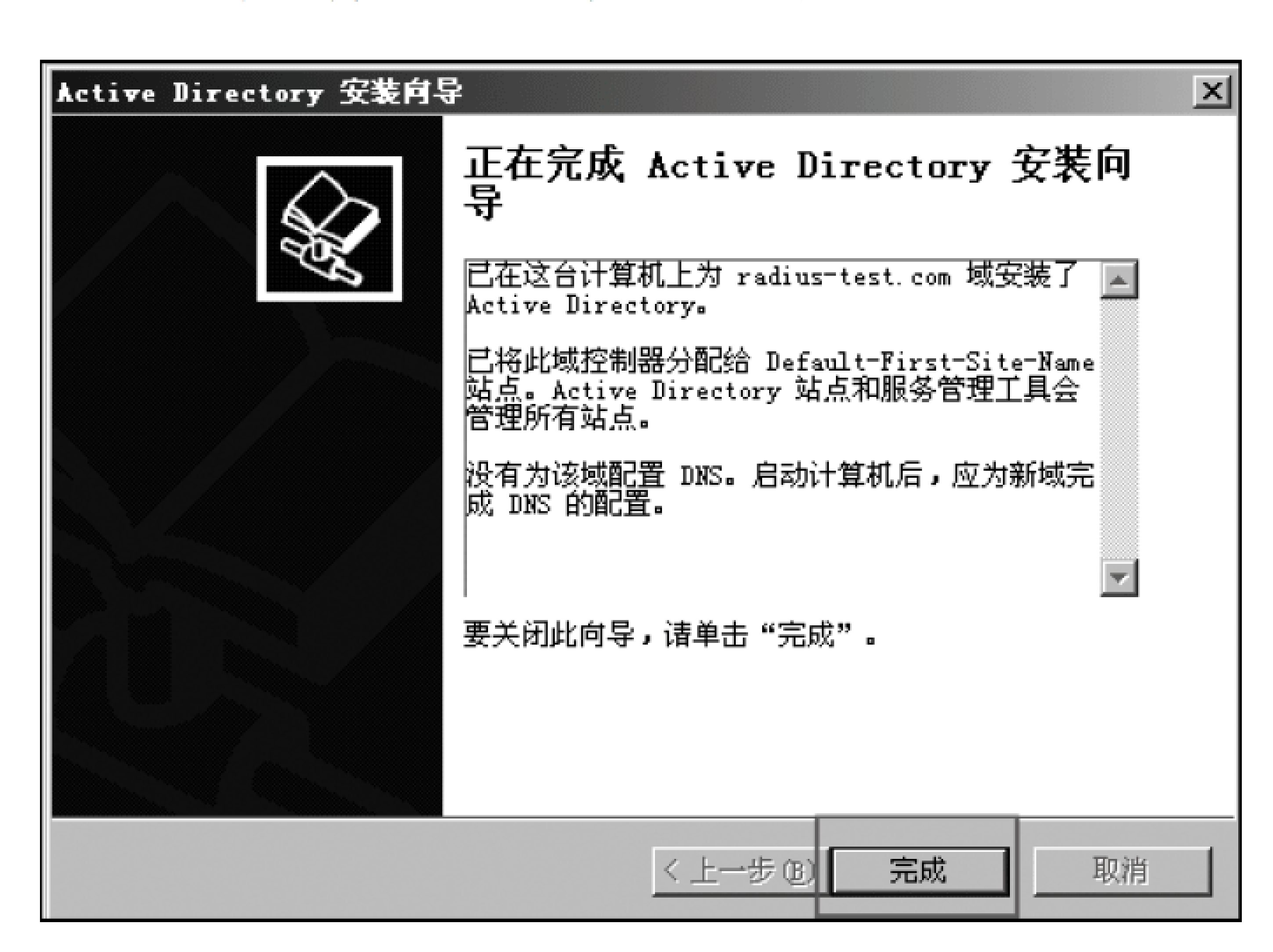


图 5-2　活动目录配置完成

如图 5-2 所示活动目录的配置完成。

如图 5-3 所示，域账户建立成功，也可自行修改相应域账户的信息。至此，完成活动目录和域控制器的配置。



图 5-3　域账户建立成功

5.3.2　DNS 服务器的安全配置

DNS 是一种组织成域层次结构的计算机和网络服务命名系统，使用域名来代替 IP 地址访问网络，它具有适用于任何网络规模，且工作不依赖于大规模的 IP 地址映射表的特征，采用分布式数据系统结构，网络运行可靠性高，在 DNS 系统中，新入网的 IP 信息可以在需要时自动广播到网络的任何一处。

所以需要先行对 DNS 服务器进行安全配置，以便在 Radius 服务中使用合格的域名。

如图 5-4 所示，完成了 DNS 配置。

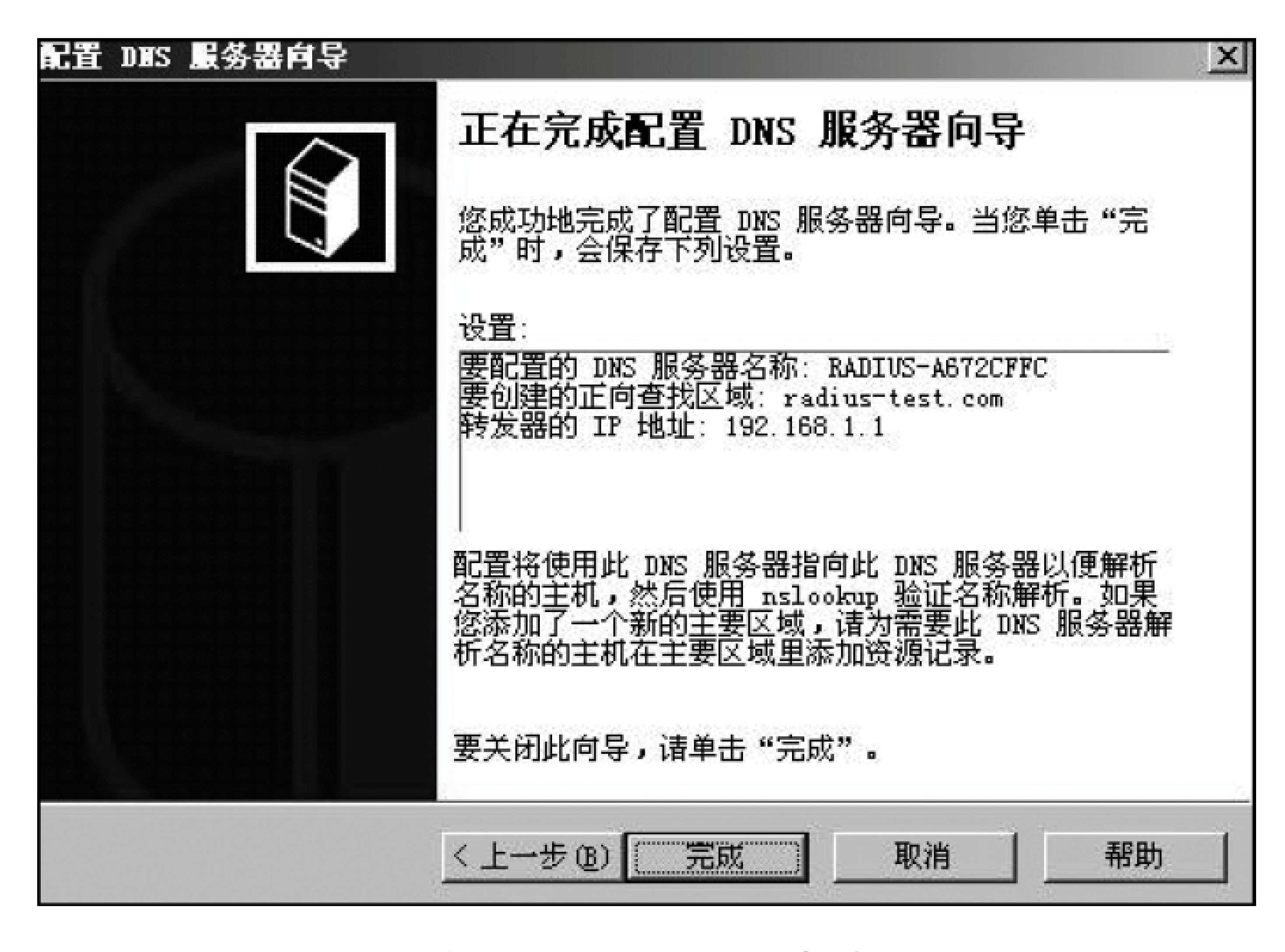


图 5-4　DNS 配置完成

5.3.3 DHCP 服务器的安全配置

如图 5-5 所示，已经成功将服务器设置为 DHCP 服务器并且进行了安全配置。

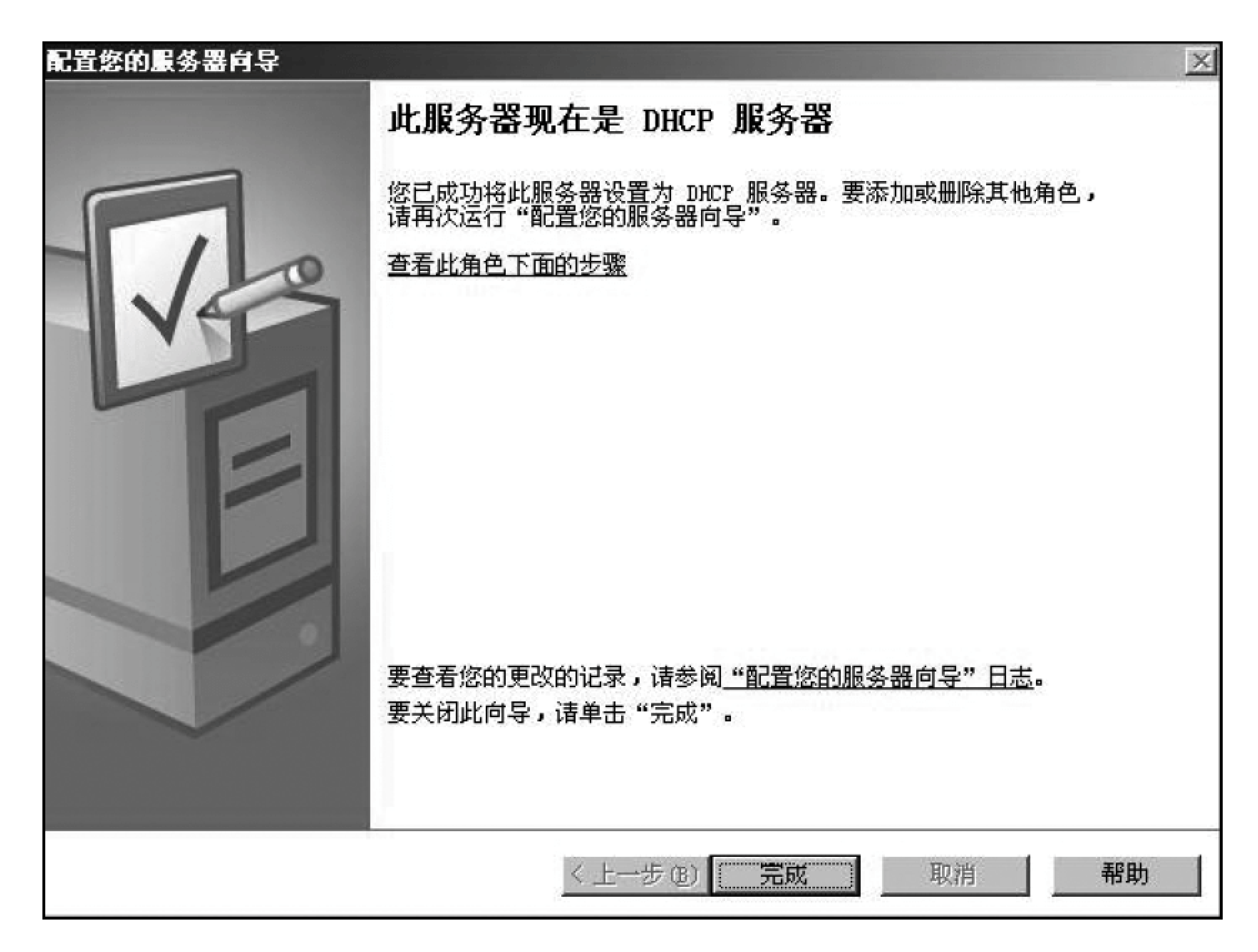


图 5-5 DHCP 的各项配置成功

完成配置后，还需选择新作用域的右键快捷菜单中的“激活”命令。

5.3.4 HTTP 服务器的安全配置

分别如图 5-6 至图 5-8 所示，依次完成了安装 IIS 6.0，创建 HTTP 站点，添加 HTTP 虚拟目录的操作，至此，就完成了 HTTP 服务器的安全配置。



图 5-6 Windows 组件向导完成



图 5-7　网站创建向导完成



图 5-8　虚拟目录创建向导完成

5.3.5　FTP 服务器的安全配置

分别如图 5-9 至图 5-11 所示，依次完成了安装 IIS 6.0，创建 FTP 站点，添加 FTP 虚拟目录的操作，至此，就完成了 FTP 服务器的安全配置。



图 5-9　完成 Windows 组件向导

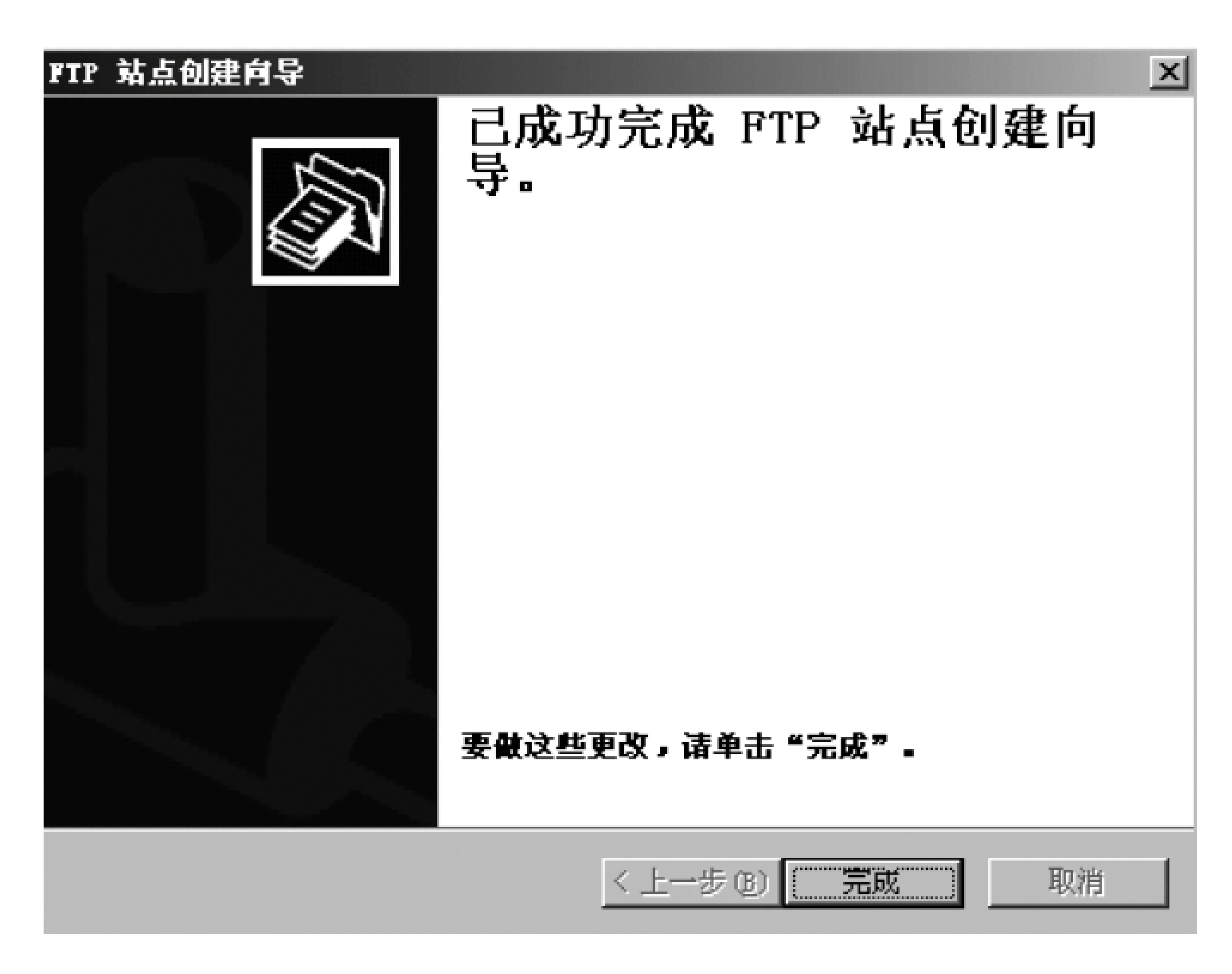


图 5-10　FTP 站点创建完成



图 5-11　成功添加了 FTP 虚拟目录

5.4　Radius 服务器和 VPN 服务器的搭建

5.4.1　Radius 服务器的搭建

如图 5-12 所示，将 IAS 服务器注册到 Active Directory。用户可以通过 Active Directory 的账户向 IAS 服务器发起连接，IAS 服务器向 Active Directory 服务器查询用户信息，确定用户是否有权连接。将 IAS 服务器添加到 Active Directory，并设置为域外控制器。接着，以管理员身份登录到 IAS 服务器，并注册 IAS 服务器到 Active Directory。

此后，可以将 IAS 服务器作为 Radius(或代理)服务器。

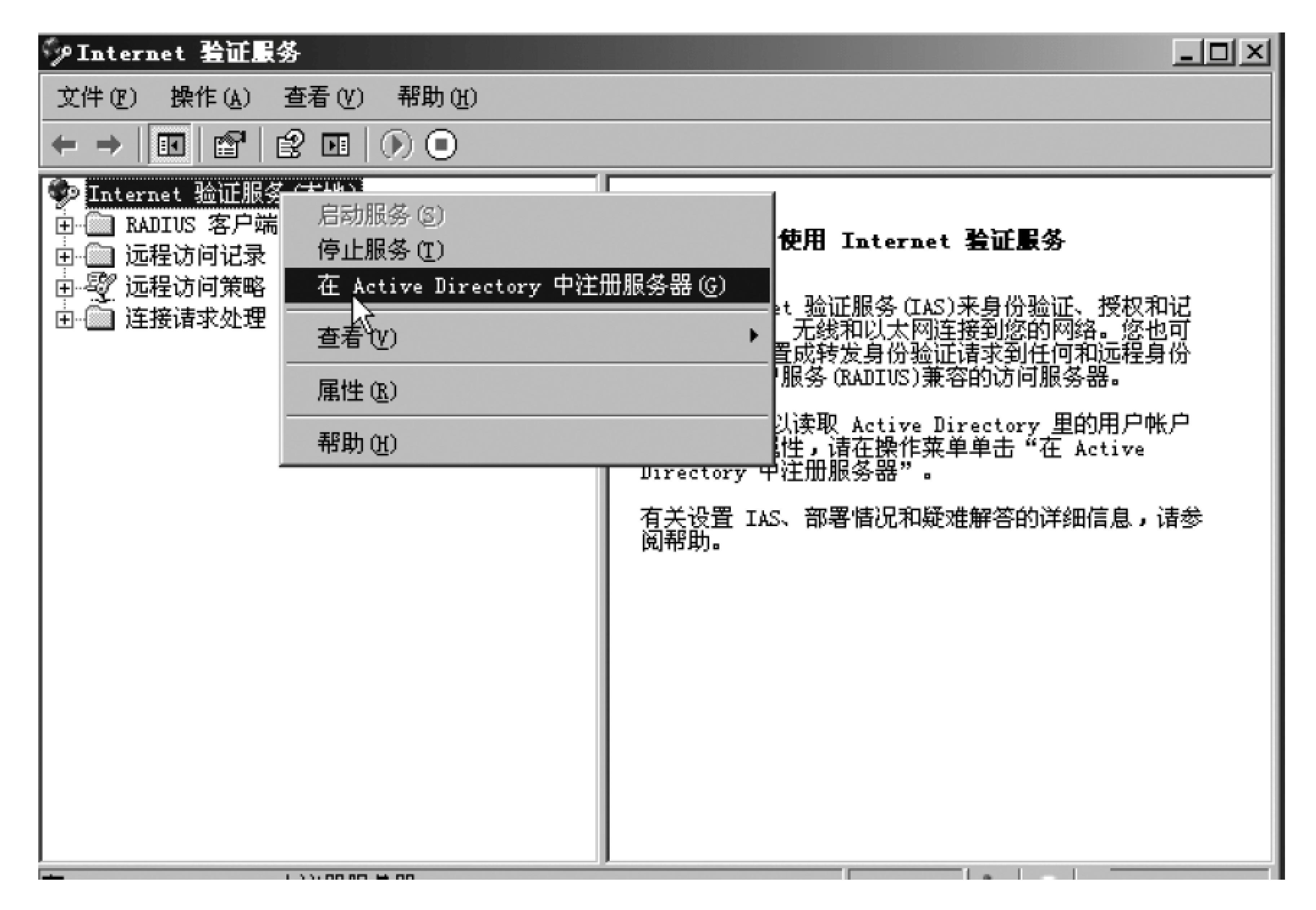


图 5-12　在 Active Directory 中注册 IAS 服务器

5.4.2　VPN 服务器的搭建

如图 5-13 所示，成功完成了 VPN 服务器的搭建。



图 5-13　VPN 服务器搭建完成

此后，可以根据个人需要，修改服务器的属性。

5.5 测试

5.5.1 测试环境

1. 硬件

- 硬盘：20GB。
- CPU：Intel Core i3，2.13GHz。
- 内存：612MB。

2. 软件

- 虚拟机：VMware Workstation 7。
- 操作系统：Windows Server 2003 SP2 Enterprise Edition。

5.5.2 测试结果

部分测试结果在前面部分是现阶段已经截图。下面是关于 DNS 服务和 FTP 服务的测试结果。

一个 DNS 域中的每台计算机都以预期完全相符合的域名为唯一标识。如前面创建活动目录时为 Windows Server 2003 选用了 radius-test.com 作为域名。域和计算机既表现为活动目录对象，又表现为 DNS 节点。所以，可以通过在开始菜单下的命令提示符中输入“ping radius-test”可测试是否能连通 DNS 域，得到如图 5-14 所示结果。

```
命令提示符
Microsoft Windows [版本 5.2.3790]
(C) 版权所有 1985-2003 Microsoft Corp.

C:\Documents and Settings\Administrator.RADIUS>ping radius-test.com

Pinging radius-test.com [192.168.1.1] with 32 bytes of data:

Reply from 192.168.1.1: bytes=32 time<1ms TTL=128
Reply from 192.168.1.1: bytes=32 time<1ms TTL=128
Reply from 192.168.1.1: bytes=32 time<1ms TTL=128
Reply from 192.168.1.1: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Documents and Settings\Administrator.RADIUS>_
```

图 5-14 测试能成功连通 DNS 域

由于已经将 TCP/IP 设置为自动获取，所以在开始菜单下的命令提示符中输入“ipconfig /all”，可以查看由 DHCP 提供服务而分配到的 IP 地址、租约等信息，如图 5-15 所示。

```
命令提示符
C:\Documents and Settings\Administrator>ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : radius-a672cffc
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No
   DNS Suffix Search List. . . . . . : localdomain

Ethernet adapter 本地连接:

   Connection-specific DNS Suffix  . : localdomain
   Description . . . . . . . . . . . : Intel(R) PRO/1000 MT Network Connection
   Physical Address. . . . . . . . . : 00-0C-29-FB-6C-C0
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   IP Address. . . . . . . . . . . . : 192.168.186.128
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.186.2
   DHCP Server . . . . . . . . . . . : 192.168.186.254
   DNS Servers . . . . . . . . . . . : 192.168.186.2
   Primary WINS Server . . . . . . . : 192.168.186.2
   Lease Obtained. . . . . . . . . . : 2012年4月14日 17:05:54
   Lease Expires . . . . . . . . . . : 2012年4月14日 17:35:54

C:\Documents and Settings\Administrator>
```

图 5-15　测试 DHCP 服务

ftp 站点设置好后，使用另一 IP 的计算机作为客户端登录：在“我的电脑”目录下输入“ftp://192.168.136.128”即可登录，得到如图 5-16 所示的主目录下的内容。



图 5-16　测试 FTP 服务器

5.6 端口扫描器的设计与实现

综合实验最终搭建出的服务器，包含 DNS、DHCP、HTTP 等多种服务，因此服务器主机同样也要打开对应的端口。本节将编写一个简单的小工具，即：端口扫描器。利用它可以检验所架设的这个多协议多服务融合的服务器上开通的服务。扫描器将利用到 Windows API 以及该平台下的网络编程相关知识。

5.6.1 端口扫描器的设计

该端口扫描器是基于 Microsoft Visual Studio 2010 平台下，由 Win32 应用程序建立的对话框程序。

1. 功能结构设计

端口扫描器功能结构流程图如图 5-17 所示。

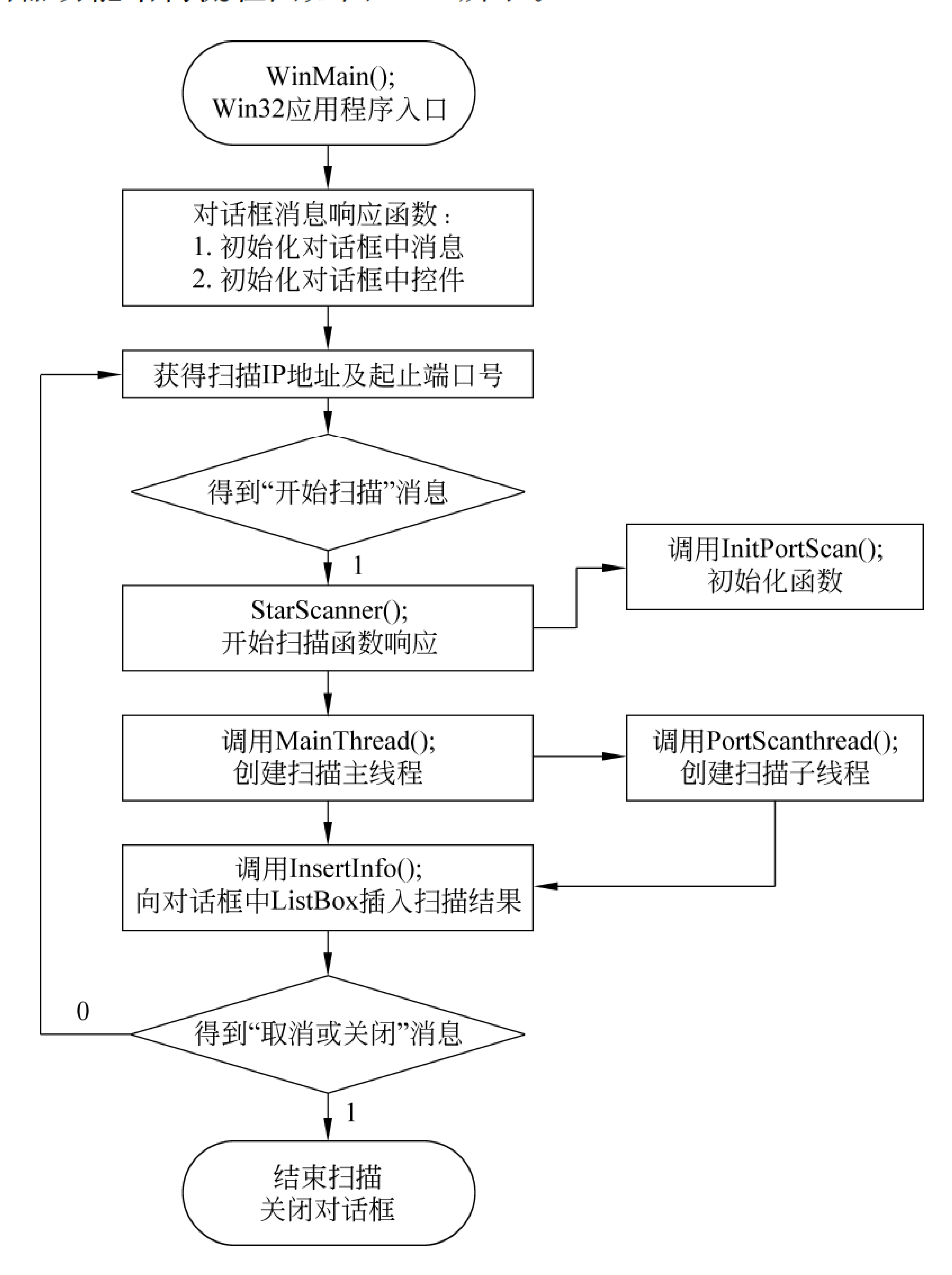


图 5-17 端口扫描器功能结构流程图

2. 界面设计

根据端口扫描器的特点以及前面功能结构的分析，设计如图 5-18 所示的界面。



图 5-18　端口扫描器界面设计图

5.6.2　端口扫描器的实现

实现步骤如下：首先在 Microsoft Visual Studio 2010 平台上建立 Win32 应用程序工程；接着新建对话框资源。在新建的对话框资源中，绘制界面、设计 UI；然后建立头文件、源文件，编写代码实现整个端口扫描器的功能。

下面给出程序实现的主要代码。

1. 端口扫描器头文件

```
////////////////////////////////////////////////////////////////////////////
//名称：Scanner.h
//功能：端口扫描器程序头文件
////////////////////////////////////////////////////////////////////////////
//包含头文件
#include<windows.h>
#include<commctrl.h>
#include<stdlib.h>
#include<winsock.h>
#include<iostream>
using namespace std;
#pragma once
#pragma comment(lib,"ComCtl32.lib")
////////////////////////////////////////////////////////////////////////////
//结构体声明部分
```

```
//////////////////////////////////////////////////////////////////////////
//主扫描线程参数结构体
struct MainThreadParam
{
    DWORD Ip;
    DWORD StartPort;
    DWORD EndPort;
    HANDLE hCopyEvent;
};

//connect 线程参数结构体
struct ThreadParam
{
    DWORD Ip;
    DWORD Port;
    HANDLE hCopyOkEvent;
    HANDLE hThreadNum;
};

//////////////////////////////////////////////////////////////////////////
//UI 以及扫描调用相关函数声明部分
//////////////////////////////////////////////////////////////////////////

//开始扫描函数
BOOL StartScanner(DWORD Ip,DWORD StartPort,DWORD EndPort);

//初始化网络
BOOL InitPortScan();

//循环调用 PortScanthread
DWORD WINAPI MainThread(LPVOID LpParam);

//端口扫描线程函数
DWORD WINAPI PortScanthread(LPVOID LpParam);

//添加扫描结果
BOOL InsertInfo( char * buf);
```

2. 端口扫描器源代码

```
//////////////////////////////////////////////////////////////////////////
//名称：Scanner.cpp
//功能：端口扫描器程序源代码
//////////////////////////////////////////////////////////////////////////
```

```
#include"resource.h"
#include "Scanner.h"
#pragma comment(lib,"comctl32.lib")
#pragma comment(lib,"WS2_32.lib")
#define MaxListNum 500

////////////////////////////////////////////////////////////////////////////
//全局变量
////////////////////////////////////////////////////////////////////////////

//List Box 控件句柄
HWND hList=NULL;
//主窗口句柄
HWND hDlg=NULL;
HINSTANCE hInst=NULL;
//ListBox 控制输出
int iIndex=MaxListNum;

////////////////////////////////////////////////////////////////////////////
//UI 相关函数声明部分
////////////////////////////////////////////////////////////////////////////

//窗口过程回调函数
LRESULT CALLBACK DlgProc(HWND hDlgMain,UINT uMsg,WPARAM wParam,LPARAM lParam);
////////////////////////////////////////////////////////////////////////////
//UI 主函数
////////////////////////////////////////////////////////////////////////////
int WINAPI WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance,LPSTR lpCmdLine,
int nShowCmd)
{
    hInst=hInstance;
    InitCommonControls();
    //窗口函数
    DialogBox(hInstance,MAKEINTRESOURCE(IDD_DIALOG),NULL,(DLGPROC)DlgProc);

    return 0;
}
////////////////////////////////////////////////////////////////////////////
//窗口过程回调函数
////////////////////////////////////////////////////////////////////////////
LRESULT CALLBACK DlgProc(HWND hDlgMain,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    hDlg=hDlgMain;
    //扫描 IP
```

```
DWORD Ip;

//起止端口
DWORD StartPort;
DWORD EndPort;

//判断扫描状态
static BOOL Flag=TRUE;

switch(uMsg)
{
    //初始化窗口消息
case WM_INITDIALOG:
    break;

    //命令消息
case WM_COMMAND:
    switch(wParam)
    {
        //"开始扫描"按钮被按下
    case IDOK:
        if(Flag)
        {
            //清空 List Box
SendMessage(GetDlgItem(hDlg,IDC_LIST_RST),LB_RESETCONTENT,NULL,NULL);

            //获取 IP
SendMessage(GetDlgItem(hDlg,IDC_IP),IPM_GETADDRESS,0,(LPARAM)&Ip);

            //获取起止端口
    StartPort=GetDlgItemInt(hDlg,IDC_EDIT_STARTPORT,NULL,FALSE);
    EndPort=GetDlgItemInt(hDlg,IDC_EDIT_ENDPORT,NULL,FALSE);
            //开始扫描
            StartScanner(Ip,StartPort,EndPort);
        }
        break;
    default:
        break;
    }
    break;
    //关闭窗口消息
case WM_CLOSE:
    EndDialog(hDlg,NULL);
    DestroyWindow(hDlg);
```

```
        break;
    default:
        break;
    }

    return 0;
}

////////////////////////////////////////////////////////////////////////////
//向 List Box 中添加信息
////////////////////////////////////////////////////////////////////////////
BOOL InsertInfo( char * buf)
{
    //其中 iIndex 是编号(从 0 开始的)可以用一个全局变量记录一共添加了 N 次(或者每次添
      加之前都获得 listbox 的 item 个数),那么 iIndex=N-1
    SendMessage(GetDlgItem(hDlg,IDC_LIST_RST),LB_RESETCONTENT,NULL,NULL);
    SendMessage(GetDlgItem(hDlg,IDC_LIST_RST),LB_ADDSTRING,0,(LPARAM)buf);
    SendMessage(GetDlgItem(hDlg,IDC_LIST_RST),LB_SETTOPINDEX,iIndex,0);
    iIndex--;

    return TRUE;
}
////////////////////////////////////////////////////////////////////////////
//开始扫描
////////////////////////////////////////////////////////////////////////////
BOOL StartScanner(DWORD Ip,DWORD StartPort,DWORD EndPort)
{
    //初始化端口扫描器
    InitPortScan();

//主扫描线程参数
    MainThreadParam param;

    //主扫描线程的"参数复制完毕"事件变量
    HANDLE hMainCopyEvent=CreateEvent(NULL,TRUE,FALSE,NULL);

    //填充参数结构体
    param.hCopyEvent=hMainCopyEvent;
    param.Ip=Ip;
    param.StartPort=StartPort;
    param.EndPort=EndPort;

    //创建主扫描线程参数
    CreateThread(NULL,0,MainThread,(LPVOID * )&param,0,NULL);
```

```
    //等待 hMainCopyEvent 变为有信号状态
    WaitForSingleObject(hMainCopyEvent,INFINITE);

    //重置 hMainCopyEvent 为无信号状态
    ResetEvent(hMainCopyEvent);

    return TRUE;
}
/////////////////////////////////////////////////////////////////////////////////
//端口器扫描初始化
/////////////////////////////////////////////////////////////////////////////////
BOOL InitPortScan()
{
    WSADATA WsaData;
    //构建 socket 版本信息
    WORD WsaVersion=MAKEWORD(2,2);
    //初始化网络
    if(WSAStartup(WsaVersion,&WsaData)! =0)
    {
        MessageBoxA(NULL,"WSAStartup Fail;",NULL,NULL);
        return FALSE;
    }

    return TRUE;
}
/////////////////////////////////////////////////////////////////////////////////
//扫描主线程
/////////////////////////////////////////////////////////////////////////////////
DWORD WINAPI MainThread(LPVOID LpParam)//APIENTRY  WINAPI
{
    MainThreadParam Param;
    //将参数复制
    MoveMemory(&Param,LpParam,sizeof(Param));
    //将 Param.hCopyEvent 设置为有信号状态
    SetEvent(Param.hCopyEvent);
    ThreadParam threadparam={0};

    //创建子线程的"参数复制完成"事件对象,并作为参数传入 PortScanthread()
    HANDLE hThreadCopyOkEvent=CreateEvent(NULL,TRUE,FALSE,NULL);
    threadparam.hCopyOkEvent=hThreadCopyOkEvent;

    //创建一个信号量对象来控制子线程的总数量 PortScanthread()
    HANDLE hThreadNum=CreateSemaphore(NULL,256,256,NULL);
```

```
    threadparam.hThreadNum=hThreadNum;

    //循环 connect
    for(DWORD Port=Param.StartPort;Port<Param.EndPort;Port++)
    {
        //等待 hThreadNum 发出信号
        DWORD WaitRes=WaitForSingleObject(hThreadNum,200);
        if(WaitRes==WAIT_OBJECT_0)
        {
            threadparam.Ip=Param.Ip;
            threadparam.Port=Port;
            CreateThread(NULL,0,PortScanthread,&threadparam,0,NULL);

            //等待其子线程发出"参数复制完毕"的信号
            WaitForSingleObject(threadparam.hCopyOkEvent,INFINITE);

            //重置 threadparam.hCopyOkEvent 为无信号状态
            ResetEvent(threadparam.hCopyOkEvent);
        }
        else if(WaitRes==WAIT_TIMEOUT)
        {
            Port--;
            continue;
        }
    }

    return 0;
}
////////////////////////////////////////////////////////////////////////
//connect 线程函数
////////////////////////////////////////////////////////////////////////
DWORD WINAPI PortScanthread(LPVOID LpParam)
{
    ThreadParam Param;
    //将参数复制
    MoveMemory(&Param,LpParam,sizeof(Param));
    //将 hCopyOkEvent 设为有信号状态来通知扫描主线程进行下一次循环
    SetEvent(Param.hCopyOkEvent);
    SOCKET Sock;
    SOCKADDR_IN SockAddr={0};
    //创建 socket
    Sock=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
    if(Sock==INVALID_SOCKET)
    {
```

```
        MessageBoxA(NULL,"INVALID_SOCKET",NULL,NULL);
    }
    //填充 IP 地址及端口信息
    SockAddr.sin_family=AF_INET;
    SockAddr.sin_addr.s_addr=htonl(Param.Ip);
    SockAddr.sin_port=htons(Param.Port);
    //将 IP 地址转换为字符串
    char * IpChar=inet_ntoa(SockAddr.sin_addr);

    char str[200];
    //开始连接
    if(connect(Sock,(SOCKADDR * )&SockAddr,sizeof(SockAddr))==0)
    {
        sprintf(str,"%s: %d 连接成功\n",IpChar,Param.Port);
    }
    else
    {
        sprintf(str,"%s: %d 连接失败\n",IpChar,Param.Port);
    }
    //添加显示信息
    InsertInfo(str);
    //释放一个信号量计数
    ReleaseSemaphore(Param.hThreadNum,1,NULL);
    //关闭 socket
    closesocket(Sock);
    return 0;
}
```

5.6.3　服务器端口开放自检验

使用已完成的端口扫描工具小程序,针对综合实验结束后的多协议多服务融合服务器进行扫描,其结果如图 5-19 所示。

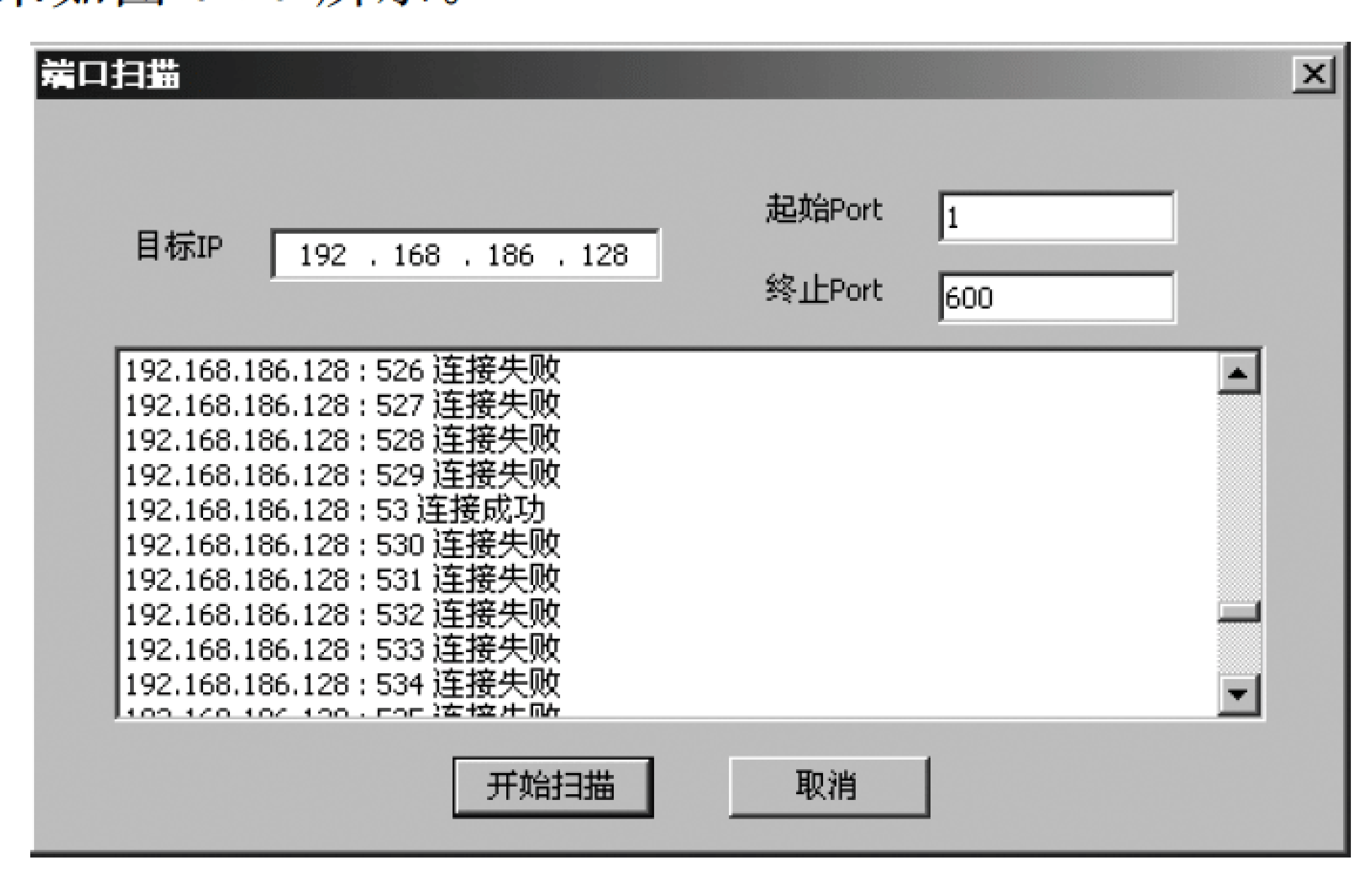


图 5-19　服务器端口扫描结果图

分别分析，连接成功的端口。为节约空间，以下扫描结果均用简略截图表示。

(1) 端口 9：丢弃

192.168.131.65 : 9 连接成功
192.168.131.65 : 90 连接失败

(2) 端口 13：时间服务

192.168.131.65 : 12 连接失败
192.168.131.65 : 13 连接成功

(3) 端口 17：每日引用

192.168.131.65 : 16 连接失败
192.168.131.65 : 17 连接成功

(4) 端口 19：字符发生器

192.168.131.65 : 18 连接失败
192.168.131.65 : 19 连接成功

(5) 端口 21：FTP 服务

192.168.116.1 : 21 连接成功
192.168.116.1 : 210 连接失败

(6) 端口 21：DNS 服务

192.168.186.128 : 53 连接成功
192.168.186.128 : 530 连接失败

(7) 端口 80：HTTP 服务

192.168.131.65 : 80 连接成功
192.168.131.65 : 81 连接失败

(8) 端口 89：SU/MIT 终端仿真开关

192.168.131.65 : 88 连接成功
192.168.131.65 : 89 连接失败

(9) 端口 139：共享资源端口

192.168.56.1 : 138 连接失败
192.168.56.1 : 139 连接成功

(10) 端口 445：NT 共享资源新端口

192.168.56.1 : 445 连接成功
192.168.56.1 : 446 连接失败

(11) 端口 464：Kpasswd

192.168.131.65 : 464 连接成功
192.168.131.65 : 465 连接失败

(12) 端口 593：http-rpc-epmap

192.168.131.65 : 592 连接失败
192.168.131.65 : 593 连接成功

通过使用工具对架设服务器 1～600 号端口的扫描，得到的结果显示，所架设的 DNS、HTTP、FTP 等服务均已开通相应端口。

5.7 小结

通过本实验，将对 Windows Server 2003 这一有别于其他 Windows 系列的操作系统有一定的了解，特别是对它的系统服务中的网络部署部分，通过实际的设置操作有更加深刻的认识。最重要的是，通过这样一步一步地配置各种网络环境，能够加深对 Radius 协议的原理、VPN 虚拟专网通信原理、DNS、DHCP、HTTP、FTP 等服务的工作原理的理解，以及对各项支持其工作的安全协议有了进一步的了解。

参考文献

[1] S. Wainner. IPSec VPN 设计[M]. 袁国忠译. 北京：人民邮电出版社，2012.
[2] 马树奇，金燕等. Windows Server 2003 从入门到精通[M]. 北京：电子工业出版社，2004.
[3] 魏茂林. Windows Server 2003 网络服务器管理与使用[M]. 北京：电子工业出版社，2007.
[4] 戴有炜. Windows Server 2003 用户管理指南[M]. 北京：清华大学出版社，2004.
[5] 赖默等. Windows Server 2008 活动目录应用指南[M]. 薛赛男，李光杰，黄湘情译. 北京：人民邮电出版社，2010.
[6] 刘晓辉. Windows Server 2008 活动目录应用实战指南[M]. 北京：清华大学出版社，2014.
[7] Hassell, Jonathan. Radius: Securing Public Access to Private Resources [M]. O'Reilly Media，2002.

第6章 IPSec综合实验

在建立IPSec安全通道之前，对等体之间需要相互认证以确定身份。Windows 2003 IPSec支持三种身份认证：Kerberos、证书和预共享密钥。只有当两个终结点（计算机）都位于同一个Windows 2003域时，Kerberos身份认证才有效。这种类型的身份认证是首选方法。如果计算机位于不同的域中，或者至少有一台计算机不在某个域中，则必须使用证书或预共享密钥。只有当每个终结点中包含一个由受信任的颁发机构签署的证书时，才能进行基于证书认证的IPSec通信。如果终结点不在同一个域中，并且无法获得证书，则预共享密钥是唯一的身份认证选择。

然而，在已有的网络安全实验教材中，IPSec身份认证实验大多数是通过"预共享密钥"的方式来进行的。因此，本章将设计两个更通用的IPSec身份认证实验："基于证书"和"基于Kerberos协议"。

6.1 实验环境和通用步骤

本章实验环境如表6-1所示。

表6-1　IPSec实验的网络和计算机配置

主　　机	操作系统	IP
域控制器DC	Windows Server 2003 (SP1)	192.168.96.3
客户机CLIENT1	Windows XP Professional (SP2)	192.168.96.4
客户机CLIENT2	Windows XP Professional (SP2)	192.168.96.5

其通用步骤为：

(1) 在控制台MMC中，添加"IP安全策略"和"IPSec监视器"两个单元，以配置IP安全策略以及观察IPSec通信状态（见图6-1）；



图6-1　IPSec实验的MMC控制台

(2) IPSec 安全策略的通用配置(见图 6-2);通过 Ping 命令来测试 IPSec 通信的安全连通性(见图 6-3)。

1. 添加一条新的IP安全策略
2. 添加IP安全规则
3. 添加IP筛选器
4. 指定新的筛选器操作的属性
5. 指派新的安全策略并进行测试

图 6-2 IPSec 实验的通用步骤

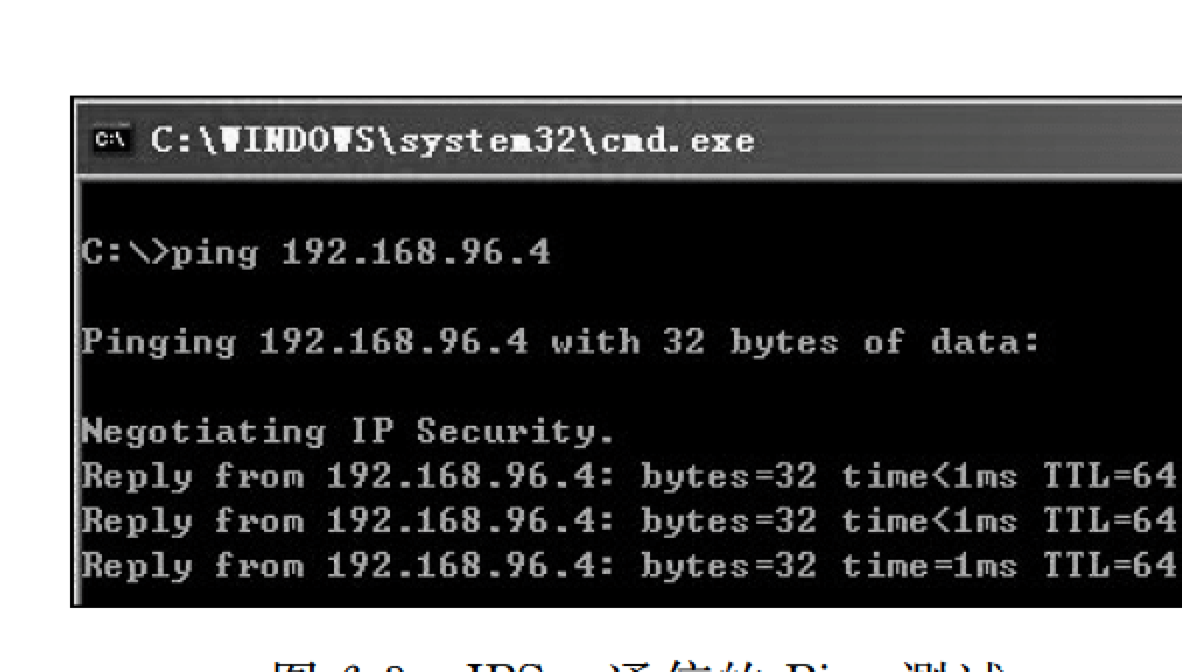


图 6-3 IPSec 通信的 Ping 测试

这 3 台计算机都是 VMWare 7 上的虚拟机,其网络模式为 Host-Only(见图 6-4)。

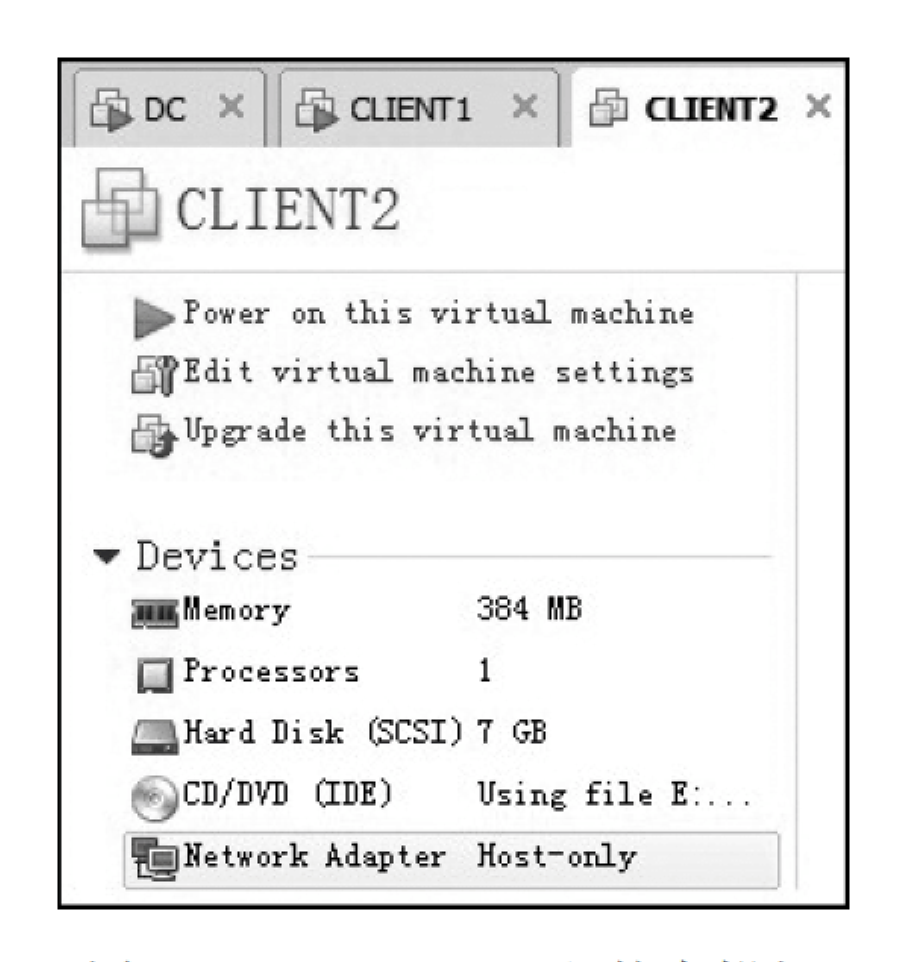


图 6-4 VMWare 7 上的虚拟机

6.2 基于 Kerberos 的 IPSec 实验

Kerberos 协议是 20 世纪 80 年代由 MIT 开发的一种分布式网络环境的身份认证协议,它基于对称密钥加密技术。Kerberos 要解决的问题是假设在一个开放的分布式环境中,工作站的用户希望访问分布在网络各处的服务器上的服务,此时服务器如何来认证用户身份并授权。

Kerberos 是 Windows 2003 唯一的身份认证机制,通过 KDC(密钥分发中心)来体现。KDC 以域为其作用范围,使用活动目录(AD)进行账号管理,并向客户端提供两个服务:认证服务(AS)和票证颁发服务(TGS)。只要安装 AD 和运行一个域控制器,Kerberos 就会安装并运行。当一个用户尝试登录时,系统就使用 Kerberos 对用户进行身份验证(见图 6-5)。

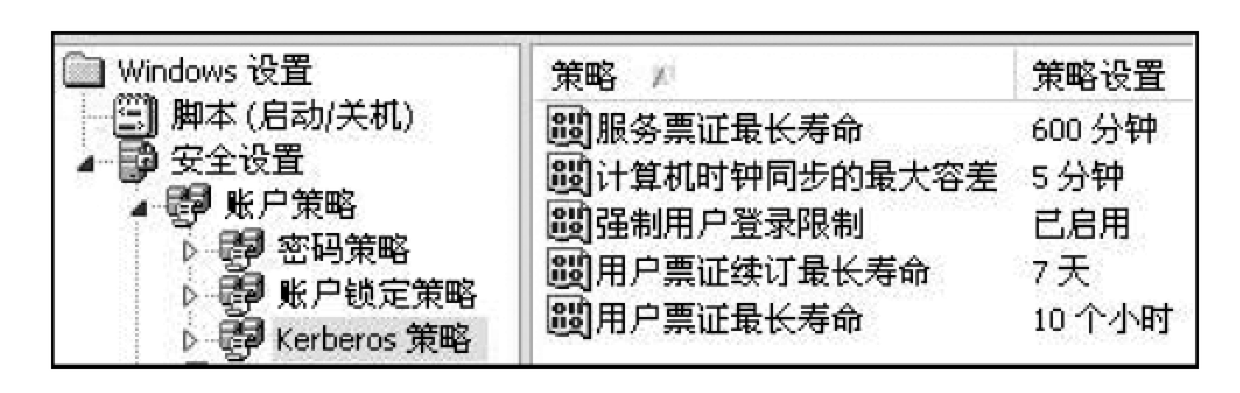


图 6-5　Kerberos 设置

本实验中，将 IP 地址为 192.168.96.3 的计算机 DC 设为域控制器，将 IP 地址为 192.168.96.4(CLIENT1)和 192.168.96.5(CLIENT2)的计算机加入该域(见图 6-6)，再按图 6-7 的配置进行实验，实验结果见图 6-8。



图 6-6　Active Directory 中的计算机

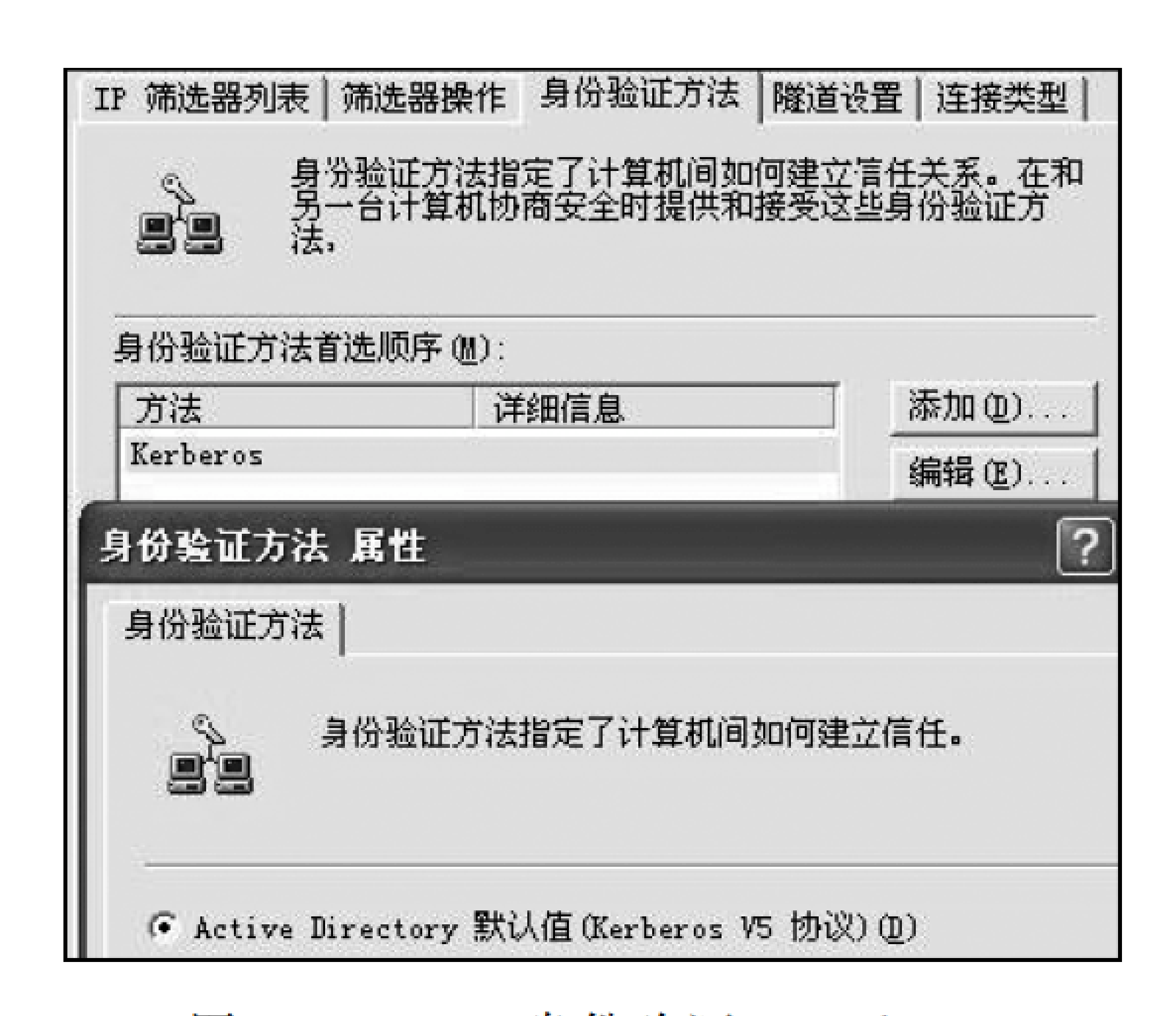


图 6-7　IPSec 身份验证—Kerberos

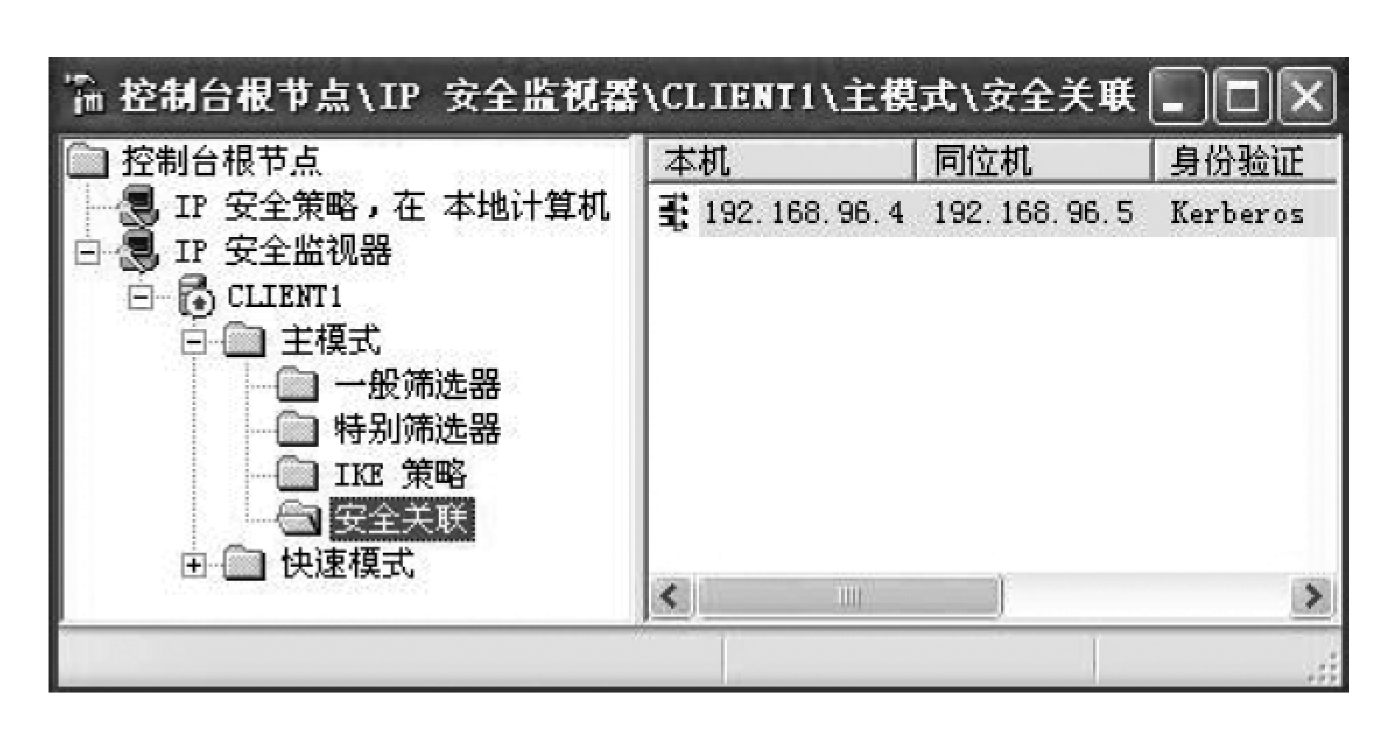


图 6-8　安全关联—Kerberos 验证

6.3 基于证书的 IPSec 实验

证书是用于身份验证的经过(权威授权机构)数字签名的声明(以文件的形式存在)。证书将公钥与保存对应私钥的实体绑定在一起,证书一般由可信的权威第三方 CA 中心(权威授权机构)颁发,CA 对其颁发证书进行数字签名,以保证所颁发证书的完整性和可鉴别性。CA 可以为用户、计算机或服务等各类实体颁发证书。

本实验中,在 IP 地址为 192.168.96.3 的计算机(DC)上设置好证书颁发机构 CA,IP 地址为 192.168.96.4(CLIENT1)和 192.168.96.5(CLIENT2)的计算机向 CA 申请证书并安装(见图 6-9 至图 6-11),具体过程参见相应资料,此处不再赘述,再按如图 6-12 的配置进行实验,实验结果如图 6-13 所示。



图 6-9　根 CA 证书



图 6-10　CLIENT1 证书

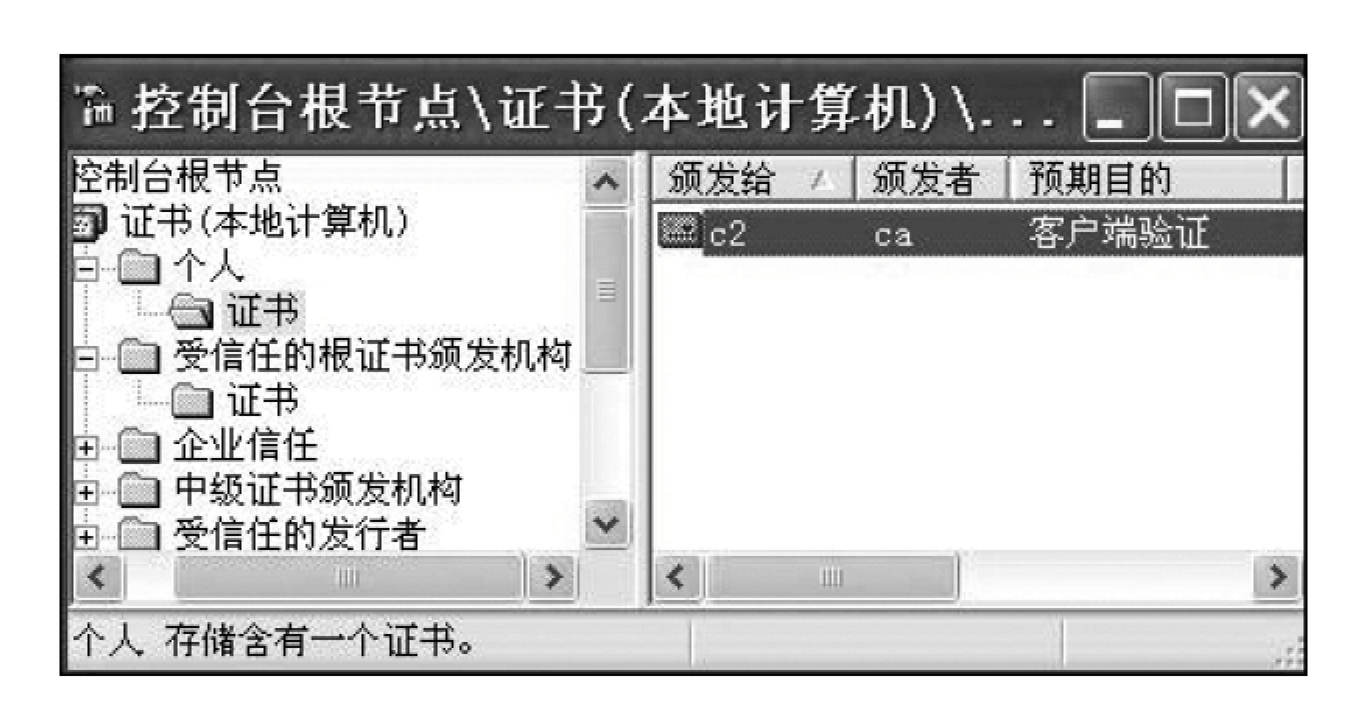


图 6-11　CLIENT2 证书

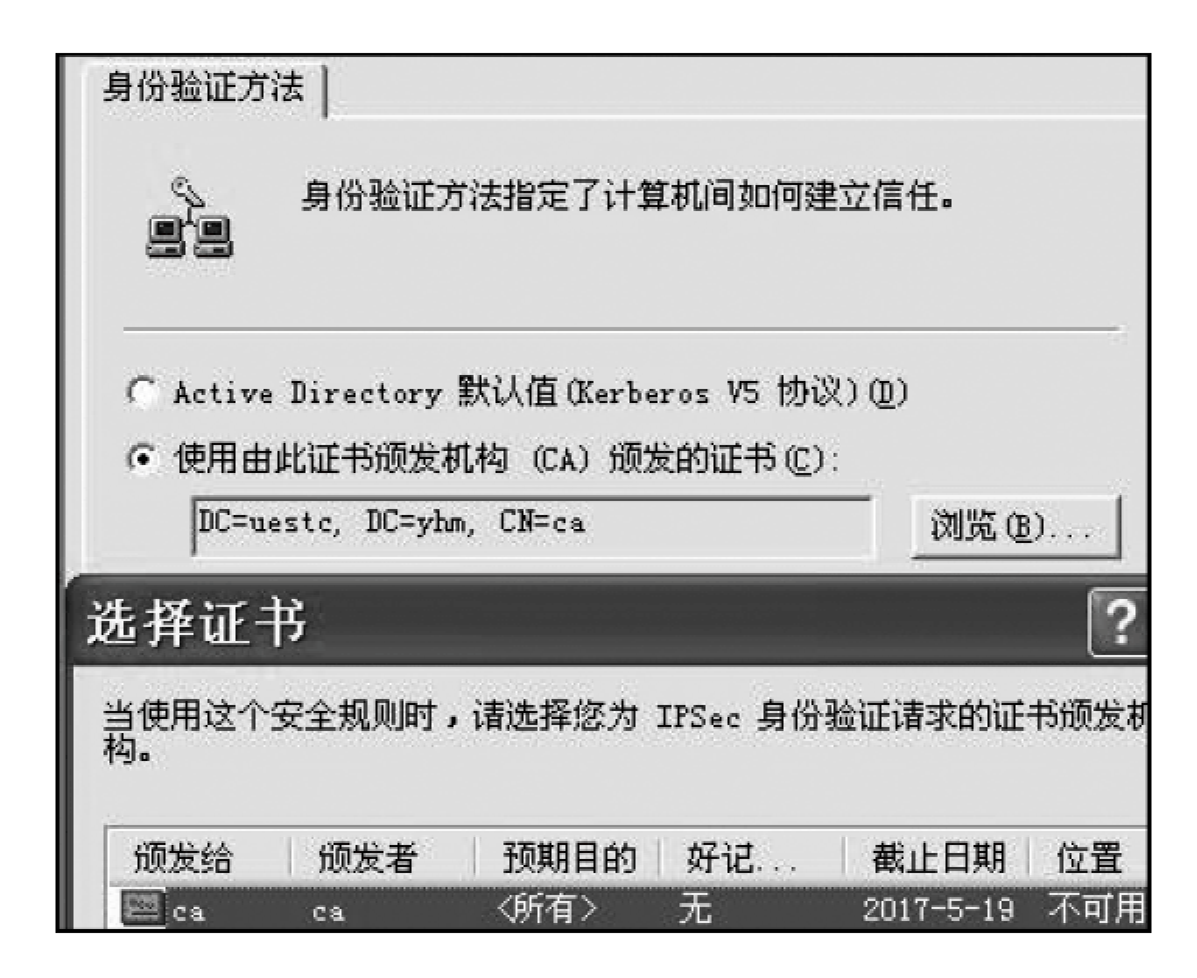


图 6-12　IPSec 身份验证-证书



图 6-13　安全关联-证书

6.4　小结

本章设计了 IPSec 身份验证的综合实验，它的身份验证方式，既包括简单的“预先共享的密钥”，一般的网络安全实验教材所设计的 IPSec 实验通常是这种；又包括比较少见的“基于 Kerberos”和“证书”，填补了在网络安全实验设计上的一个空白。本实验除了包括 IPSec、证书、Kerberos 等，还涉及 Windows 平台上各种安全设置，如活动目录、计算机加入域、防火墙、证书颁发中心 CA 等知识点，这将有助于综合掌握网络安全协议的基础知识，以及更好的培养实践工程能力。本实验设计的另一个特点是，在已有的网络实验平台的基础上，充分利用 VMWare 强大的虚拟功能，既有效地达到了实验目的，又节约了成本。

参 考 文 献

[1]　王志海等. OpenSSL 与网络信息安全：基础、结构和指令[M]. 北京：清华大学出版社，2007.
[2]　S. Wainner. IPSec VPN 设计[M]. 袁国忠译. 北京：人民邮电出版社，2012.
[3]　孙建国等. 网络安全实验教程[M]. 北京：清华大学出版社，2011.

[4] 贺思德.计算机网络信息安全与应用[M].北京：清华大学出版社，2012.
[5] 刘建伟，王育民.网络安全——技术与实践[M].北京：清华大学出版社，2005.
[6] 王煜林，田桂丰，王金恒.网络安全技术与实践[M].北京：清华大学出版社，2013.
[7] Viega，John，Messier 等. Network Security with Open SSL [M]. O'Reilly Media，2002.
[8] 吴迪，何坚，潘嵘，刘聪.基于 VMware 虚拟网络的 IPSec 实验教学[J].实验技术管理，2010(9).
[9] Doraswamy，Naganand，Harkins，Dan. IPSec：The New Security Standard for the Internet，Intranet，and Virtual Private Networks[M]，Prentice Hall，2003.

第 7 章 OpenSSL 综合实验

OpenSSL 是一个支持 SSL 认证的服务器。它是一个源码开放的自由软件，支持多种操作系统。OpenSSL 软件的目的是实现一个完整的、健壮的、商业级的开放源码工具，通过强大的加密算法来实现建立在传输层之上的安全性。OpenSSL 包含一套 SSL 协议的完整接口，应用程序应用它们可以很方便地建立起安全套接层，进而能够通过网络进行安全的数据传输。

本章要求进行 OpenSSL 综合实验，具体内容包括下面三部分：首先编译 OpenSSL 源码包，然后使用编译好的 OpenSSL 命令来创建自己的 CA，最后利用编译好的 OpenSSL 库来编程实现安全的 C/S 程序。

7.1 OpenSSL 源码包编译实验

【实验目的】

掌握 OpenSSL 源码包编译的过程，为后面的 OpenSSL 编程做好准备。

【实验内容】

下载 openssl-0.9.8zc 源码包（OpenSSL 官网 http://www.openssl.org/source/），利用 Perl 和 VS 2010 编译 OpenSSL 源码包。

【实验环境】

（1）安装 Windows XP 的计算机；

（2）Visual Studio 2010 编程环境；

（3）ActivePerl-5.20.1.2000 及以上版本（注意：如果计算机上没有安装 Perl，可以去 http://aspn.activestate.com/ASPN/Downloads/ActivePerl/下载 ActivePerl Windows 安装程序）；

（4）openssl-0.9.8zc 源码包。

【实验参考步骤】

解压 OpenSSL 源代码到 C:\openssl-0.9.8zc，然后进入到 openssl-0.9.8zc 目录，用 Administrator 身份运行 Visual Studio 2010 命令提示，然后执行如下步骤：

（1）指定 OpenSSL 编译好后的安装路径：

```
perl Configure VC-WIN32 --prefix=C:/openssl;
```

（2）创建 makefile 文件：

```
ms\do_ms
```

(3) 编译动态库：

```
nmake -f ms\ntdll.mak
```

(4) 检查上一步编译是否成功：

```
nmake -f ms\ntdll.mak test
```

(5) 安装编译后的 OpenSSL 到指定目录：

```
nmake -f ms\ntdll.mak install
```

如果 OpenSSL 源码包编译成功，则在 C:\openssl 下包含了三个子文件夹 bin、include 和 lib 以及配置文件 openssl.cnf。

7.2 使用 OpenSSL 创建 CA 实验

【实验目的】

掌握使用 OpenSSL 创建 CA 的过程。

【实验内容】

首先使用 OpenSSL 创建自己的 CA，然后该 CA 为服务器和浏览器签署证书，再将 CA 证书、服务器证书以及客户端证书安装到 Windows Server 2003，最后用 https 访问 127.0.0.1 来验证所创建 CA 的正确性。

【实验环境】

(1) 安装 Windows XP 的计算机；

(2) 安装 Windows Server 2003 的计算机；

(3) 编译好的 OpenSSL 执行程序。

【实验参考步骤】

首先需要说明的是本实验中的 CA 是安装在编译 OpenSSL 所在的 Windows XP 计算机上，由该 CA 为 Windows Server 2003 中的 IIS 网站和浏览器签署证书。而 CA 正确性的测试是在 Windows Server 2003 中 IIS 网站和浏览器之间的 SSL 通信来完成的。该实验的参考步骤如下：

(1) 在 Windows Server 2003 中安装 IIS；

(2) 在 Windows Server 2003 中获取 IIS 证书请求：

如图 7-1 所示，架设好 IIS 网站后，在"目录安全性"选项卡中单击"服务器证书"按钮，单击"下一步"按钮，选择"新建证书"选项，单击"现在准备证书请求——下一步"按钮，输入"名称"，输入"单位"和"部门"，输入"公用名称"，选择"国家"并输入"省"和"市县"并单击三次"下一步"按钮，再单击"完成"按钮，IIS 的证书请求已经获取，就是 certreq.txt。这里请牢记输入的信息。具体步骤和图示参考本书前面的 SSL 基础实验，此处不再赘述。

(3) 在 Windows XP 中进入 C:\openssl\bin 目录，并创建如下目录树：



图 7-1　服务器证书请求的获取

```
demoCA
|__certs
|__newcerts
|__private
|__crl
```

(4) 生成随机数文件：

在 private 目录下生成随机数文件.rnd(可将一个文件内容复制.rnd，例如将一个 exe 文件复制成.rnd)。

(5) 生成文本数据库文件：

demoCA 目录下手动创建一个空的文本数据库文件 index.txt。

(6) 生成证书序列号文件：

demoCA 目录下创建证书序列号文件 seria，使用文本编辑器打开，在文件中输入“01”。

(7) 生成自签名根证书：

```
openssl req -x509 -newkey rsa:1024 -keyout cakey.pem -out cacert.pem -days
3650 -config C:\openssl\openssl.cnf
```

(8) 复制文件：

```
copy cakey.pem demoCA\private
copy cacert.pem demoCA
```

(9) 用 CA 证书 cacert.pem 为 IIS 请求 certreq.txt 签发证书 server.pem：

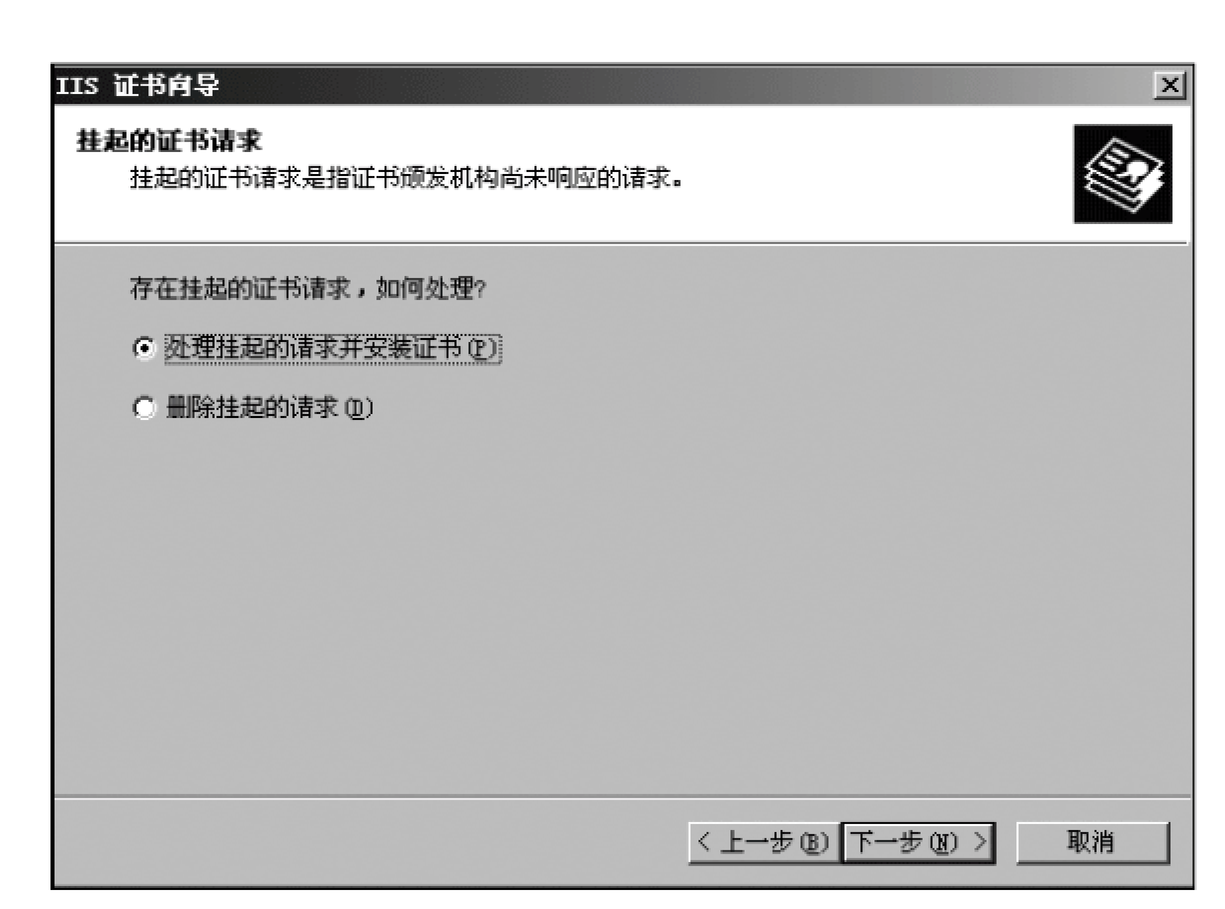
```
openssl ca -in certreq.txt -out server.pem -config C:\openssl\openssl.cnf
```

(10) 把 server. pem 转换成 x509 格式：

```
openssl x509 -in server.pem -out server.cer
```

(11) 将生成的证书 server. cer 导入到 IIS：

如图 7-2 所示，在 Windows Server 2003 中打开 IIS，在“默认网站”上右击，选择“属性”命令，在“目录安全性”选项卡中单击“服务器证书”按钮，单击“下一步”按钮，选择“处理挂起的请求并安装证书”选项并单击“下一步”按钮，正常情况下，可以看到文本框中就是 server. cer，如果不是，则单击“浏览”按钮查找并单击两次“下一步”按钮，再单击“完成”按钮。回到“目录安全性”选项卡，在“安全通信”栏目中单击“编辑”按钮，选中“要求安全通道(SSL)”复选框，再选中“要求 128 位加密”复选框，选择“要求客户端证书”，单击“确定”按钮。具体步骤和图示参考本书前面的 SSL 基础实验，此处不再赘述。



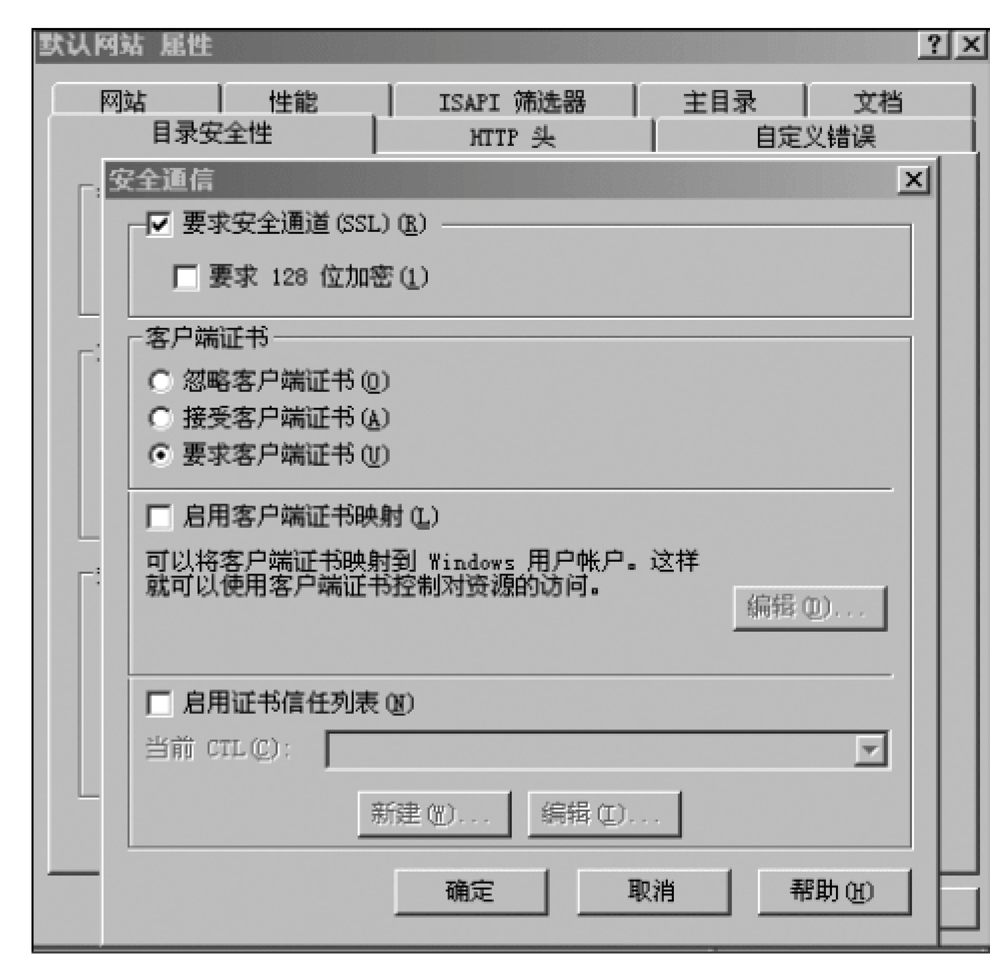


图 7-2 服务器证书的安装以及 SSL 安全通信的启用

(12) 生成客户端的密钥文件(公私钥都存放在同一个文件中)：

```
openssl genrsa -out client.key 1024
```

(13) 生成客户端证书请求文件：

```
openssl.exe req -new -key client.key -out client.csr -config C:\openssl\
openssl.cnf
```

(14) 使用 CA 根证书签名请求签名文件，生成客户端证书：

```
openssl ca -keyfile cakey.pem -cert cacert.pem -in client.csr -out client.pem
-days 365 -config C:\openssl\openssl.cnf
```

(15) 将生成的客户端证书文件转换为 pfx 格式，以利于在浏览器上导入证书：

```
openssl pkcs12 -export -in client.pem -inkey client.key -out client.pfx
```

(16) 安装客户端证书：

在 Windows Server 2003 中，选择“浏览器工具”栏中的“Internet 选项”，如图 7-3 所示将生成的 client. pfx 导入到浏览器中。



图 7-3　浏览器证书的安装

(17) 安装信任的根证书：

在 Windows Server 2003 中，把 cacert.pem 改名为 cacert.cer，双击 cacert.cer 文件，打开证书信息窗口，单击“安装证书”按钮，如图 7-4 所示进行根证书的存储设置。

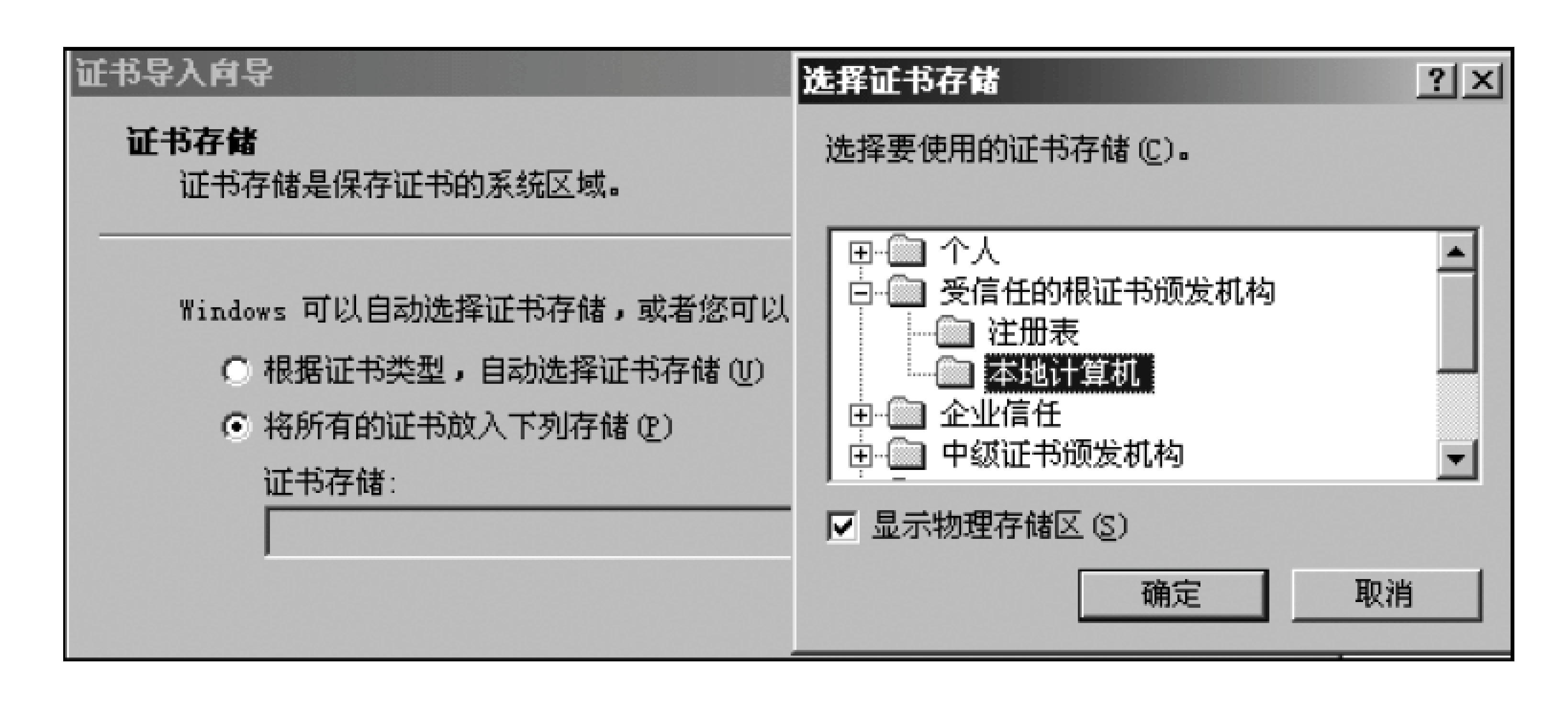


图 7-4　根证书的安装

(18) 测试 CA 的正确性：

在 Windows Server 2003 中，打开浏览器并在地址栏中输入 https://127.0.0.1，如果前面的步骤正确，则出现如图 7-5 所示的界面。

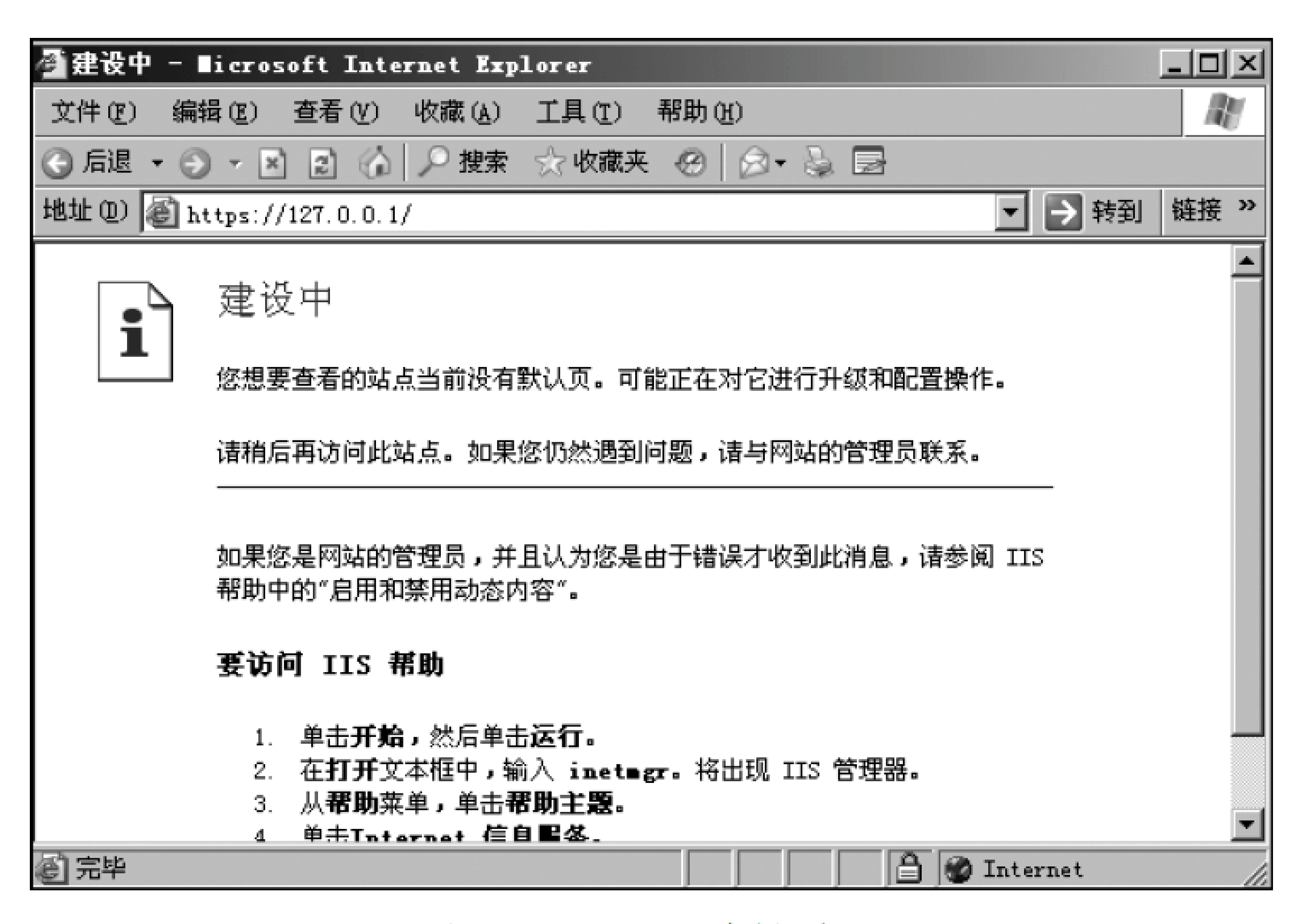


图 7-5　CA 正确性验证

7.3　OpenSSL 编程实验

【实验目的】

基于 OpenSSL 编程实现安全的 C/S 通信程序。

【实验内容】

利用编译好的 OpenSSL 库来编程实现 OpenSSL 客户端和 OpenSSL 服务器，并在两者之间进行安全的通信。

【实验环境】

(1) 安装 Windows 操作系统的计算机；

(2) Visual Studio 2010 编程环境；

(3) 编译好的 OpenSSL 库。

【实验原理】

基于 OpenSSL 的程序可以被分为两个部分：客户机和服务器，使用 SSL 协议使通信双方可以相互验证对方身份的真实性，并且能够保证数据的完整性和机密性。SSL 通信模型采用标准的 C/S 结构，除了在 TCP 层上进行传输之外，与普通的网络通信协议没有太大的区别。其过程如图 7-6 所示。

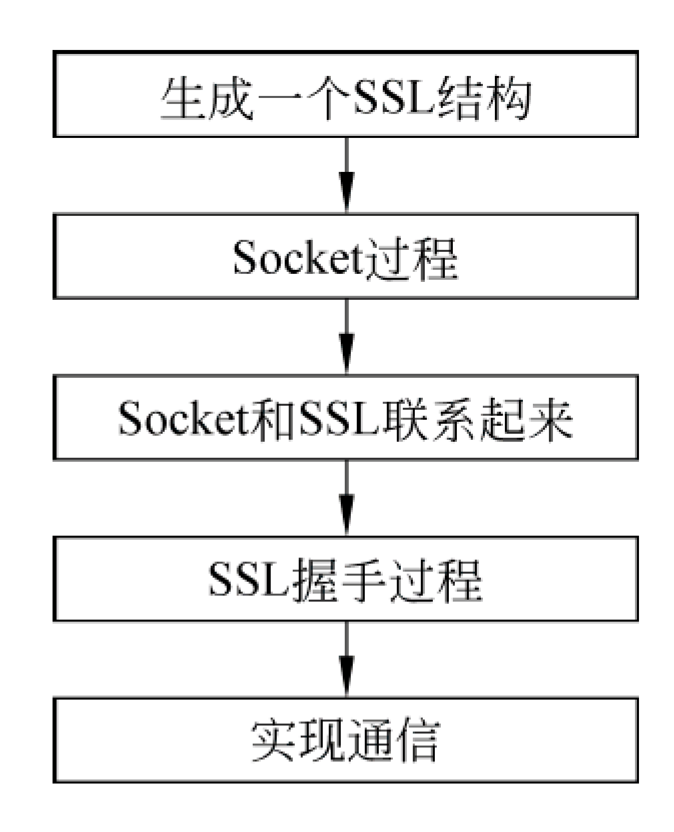


图 7-6　OpenSSL 通信过程

【实验参考步骤】

(1) 创建会话环境。

在 OpenSSL 中创建的 SSL 会话环境称为 CTX，使用不同的协议会话，其环境也是不一样的。会话环境的创建主要使用了以下几个接口函数：

```
//申请 SSL 会话环境
SSL_CTX * SSL_CTX_new(SSL_METHOD * method);
//为 SSL 会话加载用户证书
SSL_CTX_use_certificate_file(SSL_CTX * ctx,const char * file,int type);
//为 SSL 会话加载用户私钥
SSL_CTX_use_PrivateKey_file(SSL_CTX * ctx,const char * file,int type);
//在将证书和私钥加载到 SSL 会话环境之后，可以验证私钥和证书是否相符
int SSL_CTX_check_private_key(SSL_CTX * ctx);
```

(2) 建立 SSL 套接字。

SSL 套接字是建立在普通的 TCP 套接字基础之上，在建立 SSL 套接字时可使用下面的一些接口函数：

```
//申请一个 SSL 套接字
SSL * SSl_new(SSL_CTX * ctx);
//绑定读写套接字
int SSL_set_fd(SSL * ssl,int fd);
```

```
//绑定只读套接字
int SSL_set_rfd(SSL * ssl,int fd);
//绑定只写套接字
int SSL_set_wfd(SSL * ssl,int fd);
```

(3) 完成 SSL 握手。

在成功创建 SSL 套接字后，客户端和服务器分别调用下面的一些接口函数来完成 SSL 握手：

```
//客户端使用 SSL_connect( )替代传统的函数 connect( )来完成握手过程
int SSL_connect(SSL * ssl);
//服务器来使用函数 SSL_ accept ( )替代传统的函数 accept ( )来完成握手过程
int SSL_accept(SSL * ssl);
```

(4) 身份验证。

握手过程完成之后，通常需要询问通信双方的证书信息，以便进行相应的验证，这可以借助下面的函数来实现：

```
//获取对等通信方的证书
X509 * SSL_get_peer_certificate(SSL * ssl);
//获取证书所用者的名字
X509_NAME * X509_get_subject_name(X509 * a);
```

(5) 安全数据传输。

在身份验证后，就可以进行安全的数据传输了。在数据传输阶段，调用下面的一些接口函数来完成对套接字的读写操作：

```
//使用 SSL_read( )替代传统的函数 read( )
int SSL_read(SSL * ssl,void * buf,int num);
//使用 SSL_ write ( )替代传统的函数 write ( )
int SSL_write(SSL * ssl,const void * buf,int num);
```

(6) 结束 SSL 通信。

当客户端和服务器之间的数据通信完成之后，调用下面的接口函数来释放已经申请的 SSL 资源：

```
//关闭 SSL 套接字
int SSL_shutdown(SSL * ssl);
//释放 SSL 套接字
void SSl_free(SSL * ssl);
//释放 SSL 会话环境
void SSL_CTX_free(SSL_CTX * ctx);
```

附实验参考程序：

```
//服务端
```

```
#define WIN32_LEAN_AND_MEAN
#include<iostream>
#include<Windows.h>
#include<winsock2.h>
#include<openssl/rsa.h>      /* SSLeay stuff */
#include<openssl/crypto.h>
#include<openssl/x509.h>
#include<openssl/pem.h>
#include<openssl/ssl.h>
#include<openssl/err.h>
#define SERVER_PORT    5003
//certificate & key 的存放路径
//Note: 必须是全路径,否则 SSL_CTX_use_certificate_file 等函数
//      无法找到文件在 windows 平台上.
//How to:
//#privatekey.pem
//openssl.exe genrsa -out privatekey.pem 2048
//#cacert.pem
//openssl.exe req -new -x509 -key privatekey.pem -out cacert.pem -days 1095 -
config openssl.cnf
#define SERVER_CERTIFICATE "c: \\cacert.pem"
#define SERVER_KEY          "c: \\privatekey.pem"
#pragma comment(lib,"ws2_32.lib")
#pragma comment(lib,"libeay32.lib")
#pragma comment(lib,"ssleay32.lib")
int main()
{
    ///////////////////
    //初始化//
    ///////////////////
    SSL_CTX* ctx;
    SSL_METHOD *meth;
    SSL_load_error_strings();
    SSLeay_add_ssl_algorithms();
    meth=(SSL_METHOD *)SSLv23_server_method();
    ctx=SSL_CTX_new (meth);
    if (!ctx) {
        ERR_print_errors_fp(stderr);
        std::cout<<"SSL_CTX_new error."<<std::endl;
        return -1;
    }
    if (SSL_CTX_use_certificate_file(ctx,SERVER_CERTIFICATE,SSL_FILETYPE_
PEM)<=0) {
        ERR_print_errors_fp(stderr);
```

```
            std::cout<<"SSL_CTX_use_certificate_file error."<<std::endl;
            return -1;
        }
        if (SSL_CTX_use_PrivateKey_file(ctx,SERVER_KEY,SSL_FILETYPE_PEM)<=0) {
            ERR_print_errors_fp(stderr);
            std::cout<<"SSL_CTX_use_PrivateKey_file error."<<std::endl;
            return -1;
    if (!SSL_CTX_check_private_key(ctx)) {
            ERR_print_errors_fp(stderr);
            std::cout<<"SSL_CTX_check_private_key error."<<std::endl;
            return -1;
        }
        ////////////////////////////////////////////
        //建立原始的 TCP 连接//
        ///////////////////////////////////////////
        WSADATA wsaData;
        SOCKET listen_socket;
        SOCKET accept_socket;
        struct sockaddr_in addr_server;
        struct sockaddr_in addr_client;
        int addr_client_len;
        int ret=WSAStartup(MAKEWORD(2,2),&wsaData);
        if (ret !=0) {
            std::cout<<"WSAStartup error."<<std::endl;
            return -1;
        }
        listen_socket=socket (AF_INET,SOCK_STREAM,0);
        if(listen_socket==INVALID_SOCKET) {
            std::cout<<"socket error."<<std::endl;
            return -1;
        }
        memset (&addr_server,0,sizeof(addr_server));
        addr_server.sin_family        =AF_INET;
        addr_server.sin_addr.S_un.S_addr=INADDR_ANY;
        addr_server.sin_port          =htons (SERVER_PORT);

        ret=bind(listen_socket,(struct sockaddr *)&addr_server,sizeof(addr_
    server));
        if(ret==SOCKET_ERROR) {
            std::cout<<"bind error."<<std::endl;
            return -1;
    }
        ret=listen (listen_socket,5);
        if(ret==SOCKET_ERROR) {
```

```
        std::cout<<"listen error."<<std::endl;
        return -1;
    }
    addr_client_len=sizeof(addr_client);
    accept_socket=accept (listen_socket, (struct sockaddr * ) &addr_client,
&addr_client_len);
if(accept_socket==INVALID_SOCKET) {
        std::cout<<"accept error."<<std::endl;
        return -1;
    }
    closesocket(listen_socket);
    std::cout<<" Connection from "<<addr_client.sin_addr.S_un.S_addr<<":"
<<addr_client.sin_port<<std:::    endl;
    //////////////////////////////////////////////////////////////////
    //TCP 连接已经建立,执行 Server SSL//
    //////////////////////////////////////////////////////////////////
    SSL *     ssl;
    X509 *    client_certificate;
    char *    str;
    ssl=SSL_new (ctx);
    if(ssl==NULL) {
        std::cout<<"SSL_new error."<<std::endl;
        return -1;
    }
    SSL_set_fd (ssl,accept_socket);
    ret=SSL_accept (ssl);
    if(ret==-1) {
        std::cout<<"SSL_accept error."<<std::endl;
        return -1;
    }
    //获取 cipher
    std::cout<<"SSL connection using: "<<SSL_get_cipher(ssl)<<std::endl;
    //获取客户端的证书
    client_certificate=SSL_get_peer_certificate (ssl);
    if (client_certificate !=NULL) {
        std::cout<<"Client certificate:"<<std::endl;
        str=X509_NAME_oneline (X509_get_subject_name (client_certificate),0,0);
        if(str==NULL) {
            std::cout<<"X509_NAME_oneline error."<<std::endl;
        } else {
            std::cout<<"subject: "<<str<<std::endl;
            OPENSSL_free (str);
        }
        str=X509_NAME_oneline (X509_get_issuer_name (client_certificate),0,0);
```

```
        if(str==NULL) {
            std::cout<<"X509_NAME_oneline error."<<std::endl;
        } else {
            std::cout<<"issuer: "<<str<<std::endl;

            OPENSSL_free (str);
        }
        X509_free (client_certificate);
    } else {
        std::cout<<"Client does not have certificate. "<<std::endl;
    }
    //////////////////////////
    // 数据交换//
    //////////////////////////
    char     buf [4096];
    ret=SSL_read (ssl,buf,sizeof(buf) -1);
    if(ret==-1) {
        std::cout<<"SSL_read error."<<std::endl;
        return -1;
    }
    buf[ret]='\0';
    std::cout<<buf<<std::endl;
    ret=SSL_write (ssl,"I hear you.",strlen("I hear you."));
    if(ret==-1) {
        std::cout<<"SSL_write error."<<std::endl;
        return -1;
    }
    //////////////////
    //Cleanup//
    //////////////////
    closesocket(accept_socket);
    SSL_free (ssl);
    SSL_CTX_free (ctx);
    WSACleanup();
    system("pause");
    return 0;
}
//客户端
#define WIN32_LEAN_AND_MEAN
#include<iostream>
#include<winsock2.h>
#include<openssl/rsa.h>      /* SSLeay stuff */
#include<openssl/crypto.h>
#include<openssl/x509.h>
```

```
#include<openssl/pem.h>
#include<openssl/ssl.h>
#include<openssl/err.h>
#define SERVER_IP     "127.0.0.1"
#define SERVER_PORT    5003
#pragma comment(lib,"ws2_32.lib")
#pragma comment(lib,"libeay32.lib")
#pragma comment(lib,"ssleay32.lib")
int main()
{
    int ret;
    //////////////////
    //初始化//
    //////////////////
    SSL_CTX * ctx;
    SSL_METHOD * meth;
    SSL_load_error_strings();
    SSLeay_add_ssl_algorithms();
    meth= (SSL_METHOD * )SSLv23_client_method();
    ctx=SSL_CTX_new (meth);
    if (!ctx) {
        ERR_print_errors_fp(stderr);
        std::cout<<"SSL_CTX_new error."<<std::endl;
        return -1;
    }
    ////////////////////////////////////////
    //建立原始的 TCP 连接//
    ////////////////////////////////////////
    WSADATA wsaData;
    SOCKET client_socket;
    struct sockaddr_in addr_server;
    ret=WSAStartup(MAKEWORD(2,2),&wsaData);
    if (ret !=0) {
        std::cout<<"WSAStartup error."<<std::endl;
        return -1;
    }
    client_socket=socket (AF_INET,SOCK_STREAM,0);
    if(client_socket==INVALID_SOCKET) {
        std::cout<<"socket error."<<std::endl;
        return -1;
    }
    memset (&addr_server,0,sizeof(addr_server));
    addr_server.sin_family        =AF_INET;
```

```
    addr_server.sin_addr.S_un.S_addr=inet_addr(SERVER_IP);
    addr_server.sin_port          =htons (SERVER_PORT);
    ret=connect(client_socket,(struct sockaddr * ) &addr_server,sizeof(addr_
server));
    if(client_socket==SOCKET_ERROR) {
        std::cout<<"connect error."<<std::endl;
        return -1;
    }
    //////////////////////////////////////////////////////////////////
    //TCP 连接已经建立,执行 Client SSL//
    //////////////////////////////////////////////////////////////////
    SSL *     ssl;
    X509 *    server_certificate;
    char *    str;

    ssl=SSL_new (ctx);
    if(ssl==NULL) {
        std::cout<<"SSL_new error."<<std::endl;
        return -1;
    }
    SSL_set_fd (ssl,client_socket);
    ret=SSL_connect (ssl);
    if(ret==-1) {
        std::cout<<"SSL_accept error."<<std::endl;
        return -1;
    }
    //接下来的获取密码和获取服务器端证书是可选的,不会影响数据交换
    //获取 cipher
    std::cout<<"SSL connection using: "<<SSL_get_cipher(ssl)<<std::endl;
    //获取服务器端的证书
    server_certificate=SSL_get_peer_certificate (ssl);
    if(server_certificate !=NULL) {
        std::cout<<"Server certificate:"<<std::endl;
        str=X509_NAME_oneline (X509_get_subject_name (server_certificate),0,0);
        if(str==NULL) {
            std::cout<<"X509_NAME_oneline error."<<std::endl;
        } else {
            std::cout<<"subject: "<<str<<std::endl;
            OPENSSL_free (str);
        }
        str=X509_NAME_oneline (X509_get_issuer_name (server_certificate),0,0);
        if(str==NULL) {
            std::cout<<"X509_NAME_oneline error."<<std::endl;
```

```
        } else {
            std::cout<<"issuer: "<<str<<std::endl;
            OPENSSL_free (str);
        }
        X509_free (server_certificate);
    } else {
        std::cout<<"Server does not have certificate. we sould Esc!"<<std::
endl;
        return -1;
    }
    //////////////////////////
    // 数据交换//
    //////////////////////////
    char    buf [4096];
    ret=SSL_write (ssl,"My name is Yang",strlen("My name is Yang"));
    if(ret==-1) {
        std::cout<<"SSL_write error."<<std::endl;
        return -1;
    }
    ret=SSL_read (ssl,buf,sizeof(buf) -1);
    if(ret==-1) {
        std::cout<<"SSL_read error."<<std::endl;
        return -1;
    }
    buf[ret]='\0';
    std::cout<<buf<<std::endl;
    SSL_shutdown(ssl); /* send SSL/TLS close_notify */
    //////////////////
    //Cleanup//
    //////////////////
    closesocket(client_socket);
    SSL_free (ssl);
    SSL_CTX_free (ctx);
    WSACleanup();
    return 0;
```

7.4　小结

通过本实验，首先了解 OpenSSL 源码包，掌握 OpenSSL 源码包编译的过程；然后，掌握使用 OpenSSL 创建 CA 的过程；最重要的是，使用基于 OpenSSL 编程实现安全的 C/S 通信程序，为以后 OpenSSL 的编程有很大的帮助。

参考文献

[1] 诸葛建伟.网络攻防技术与实践[M].北京:电子工业出版社,2011.
[2] 张玉清.网络攻击与防御技术[M].北京:清华大学出版社,2011.
[3] 贺思德.计算机网络信息安全与应用[M].北京:清华大学出版社,2012.
[4] 刘建伟,王育民.网络安全——技术与实践[M].北京:清华大学出版社,2005.
[5] 王煜林,田桂丰,王金恒.网络安全技术与实践[M].北京:清华大学出版社,2013.
[6] Viega John,Messier 等.Network Security with Open SSL [M]. O'Reilly Media,2002.
[7] Oppliger,Rolf.SSL and TLS: Theory and Practice [M]. Artech House Publishers,2009.
[8] Rescorla,Eric. SSL and TLS: Designing and Building Secure Systems [M]. Addison-Wesley Professional,2000.

第 8 章 VPN 综合实验

虚拟专用网络(Virtual Private Network,VPN)指的是在公用网络上建立专用网络的技术。整个 VPN 网络的任意两个节点之间的连接并没有传统专网所需的端到端的物理链路,而是架构在公用网络服务商所提供的网络平台上。VPN 涉及到隧道技术、加解密技术、身份验证链接技术等。因此,了解和应用 VPN 也必将成为网络安全课程中的重要内容。探讨如何开展 VPN 实验,对提高学生计算机网络安全实践操作能力具有现实的指导意义。由于 Windows 2003 并不直接支持基于 SSL 的 VPN,本章将在 Windows 2008 平台下设计 VPN 实验。

Windows Server 2008 的 VPN 有三种类型:PPTP、L2TP/IPSec 和 SSTP。然而,在已有的网络安全实验教材中,VPN 实验是基于 PPTP 协议的。因此,本章将设计两个更通用的 VPN 实验:“基于 L2TP/IPSec”和“基于 SSTP”。

8.1 VPN 简介

8.1.1 VPN 的功能

VPN 属于远程访问技术,简单地说就是利用公网链路架设私有网络。例如,公司员工出差到外地,他想访问企业内网的服务器资源,这种访问就属于远程访问。怎么才能让外地员工访问到内网资源呢?VPN 的解决方法是在内网中架设一台 VPN 服务器,VPN 服务器有两块网卡:一块连接内网,一块连接公网。外地员工在当地连上互联网后,通过互联网找到 VPN 服务器,然后利用 VPN 服务器作为跳板进入企业内网。为了保证数据安全,VPN 服务器和客户机之间的通信数据都进行了加密处理。有了数据加密,就可以认为数据是在一条专用的数据链路上进行安全传输,就如同专门架设了一个专用网络一样。但实际上 VPN 使用的是互联网上的公用链路,因此只能称为虚拟专用网。即:VPN 实质上就是利用加密技术在公网上封装出一个数据通信隧道。有了 VPN 技术,用户无论是在外地出差还是在家中办公,只要能上互联网就能利用 VPN 非常方便地访问内网资源,这使得 VPN 在企业中应用得如此广泛。

虚拟专用网的提出就是来解决如下问题:

(1) 使用 VPN 可降低成本——通过公用网来建立 VPN,就可以节省大量的通信费用,而不必投入大量的人力和物力去安装和维护 WAN(广域网)设备和远程访问设备。

(2) 传输数据安全可靠——虚拟专用网产品均采用加密及身份验证等安全技术,保证连接用户的可靠性及传输数据的安全和保密性。

(3) 连接方便灵活——用户想与合作伙伴联网,如果没有虚拟专用网,双方的信息技术部门就必须协商如何在双方之间建立租用线路或帧中继线路,有了虚拟专用网之后,只需双方配置安全连接信息即可。

(4) 完全控制——虚拟专用网使用户可以利用 ISP 的设施和服务,同时又完全掌握着自己网络的控制权。用户只利用 ISP 提供的网络资源,对于其他的安全设置、网络管理变化可由自己管理。在企业内部也可以自己建立虚拟专用网。

8.1.2 VPN 的技术

1. 隧道技术

实现 VPN 的最关键部分是在公网上建立虚信道,而建立虚信道是利用隧道技术实现的,IP 隧道的建立可以是在链路层和网络层。第二层隧道主要是 PPP 连接,如 PPTP、L2TP,其特点是协议简单,易于加密,适合远程拨号用户;第三层隧道是 IP in IP,如 IPSec,其可靠性及扩展性优于第二层隧道,但没有前者简单直接。

2. 隧道协议

隧道是利用一种协议传输另一种协议的技术,即用隧道协议来实现 VPN 功能。为创建隧道,隧道的客户机和服务器必须使用同样的隧道协议。

(1) PPTP(点到点隧道协议)是一种用于让远程用户拨号连接到本地的 ISP,通过因特网安全远程访问公司资源的新型技术。它能将 PPP(点到点协议)帧封装成 IP 数据包,以便在基于 IP 的互联网上进行传输。PPTP 使用 TCP(传输控制协议)连接创建、维护与终止隧道,并使用 GRE(通用路由封装)将 PPP 帧封装成隧道数据。被封装后的 PPP 帧的有效载荷可以被加密或者压缩或者同时被加密与压缩。

(2) L2TP 协议:L2TP 是 PPTP 与 L2F(第二层转发)的一种综合,它是由思科公司所推出的一种技术。

(3) IPSec 协议:是一个标准的第三层安全协议,它是在隧道外面再封装,保证了在传输过程中的安全。IPSec 的主要特征在于它可以对所有 IP 级的通信进行加密。

3. 加解密技术和密钥管理技术

加解密技术是数据通信中一现较成熟的技术,VPN 可直接利用现有技术实现加解密。密钥管理技术的主要任务是如何在公用数据网上安全地传递密钥而不被窃取。

4. 使用者与设备身份认证技术

使用者与设备认证及时最常用的是使用者名称与密码或卡片式认证等方式。

8.2 Windows Server 2008 的 VPN 简介

Windows Server 2008 所支持的 VPN 有三种类型:PPTP、L2TP/IPSec 和 SSTP,其体系结构如图 8-1 所示。其中,SSTP(Secure Socket Tunneling Protocol)是微软提供的新一代的虚拟专用网(VPN)技术。相对于 PPTP 和 L2TP/IPSec 隧道协议,其流量通过

防火墙的困难性，SSTP 能够使流量更容易通过防火墙，在拥有最大弹性发挥的同时，又确保信息安全达到了一定程度。

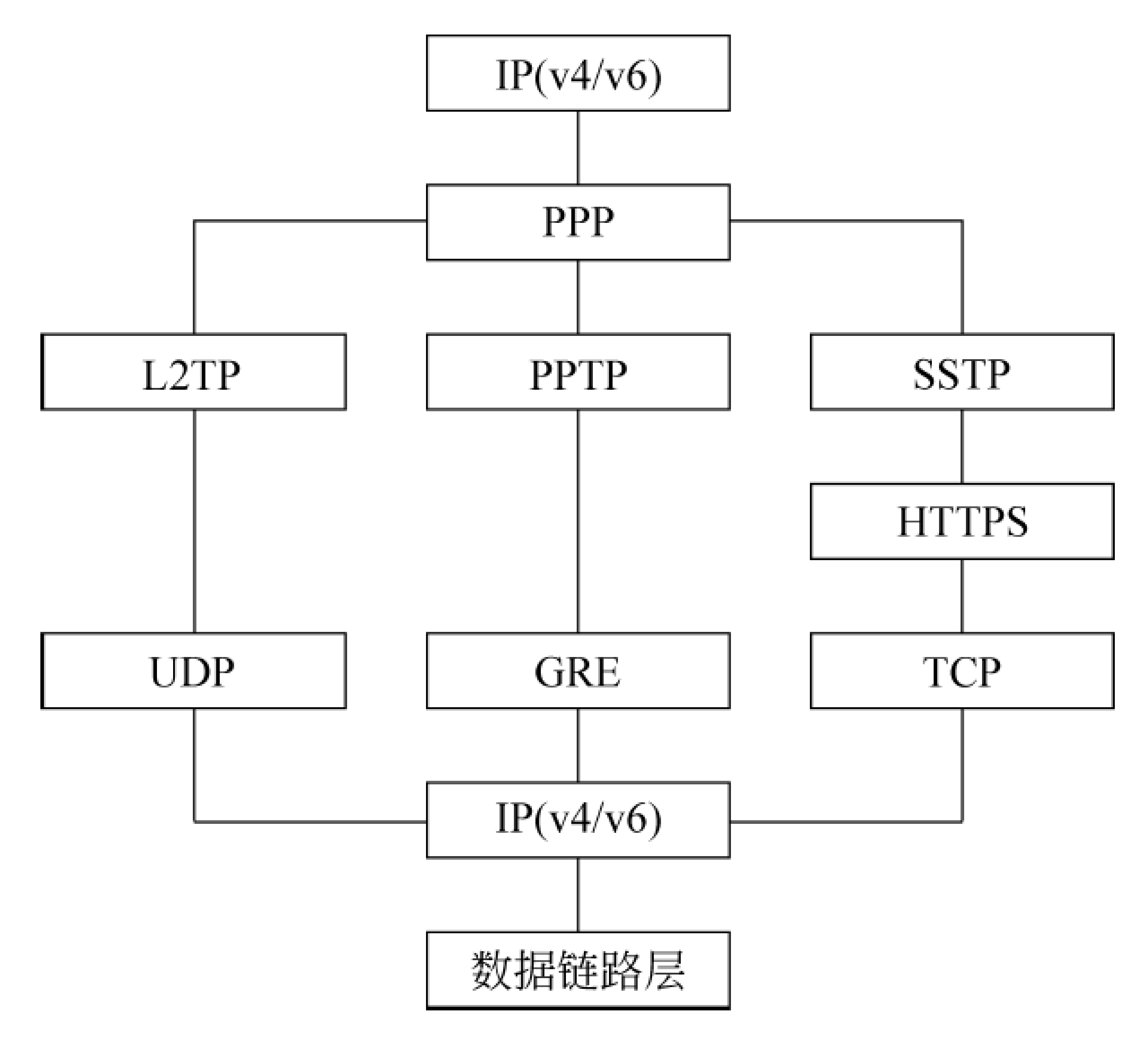


图 8-1　Windows Server 2008 的 VPN 的体系结构

8.3　基于虚拟机的 VPN 的综合实验设计

基于 PPTP 的 VPN 实验在大多数网络安全协议的实验教材中均能找到，因此这里仅简单介绍，用它来展示实验环境和通用步骤。基于 L2TP　over IPSec 或基于 SSTP 的 VPN 却很少见到，本节将设计这两个 VPN 实验。

8.3.1　实验环境和通用步骤

实验的网络拓扑如图 8-2 所示，其通用步骤为：

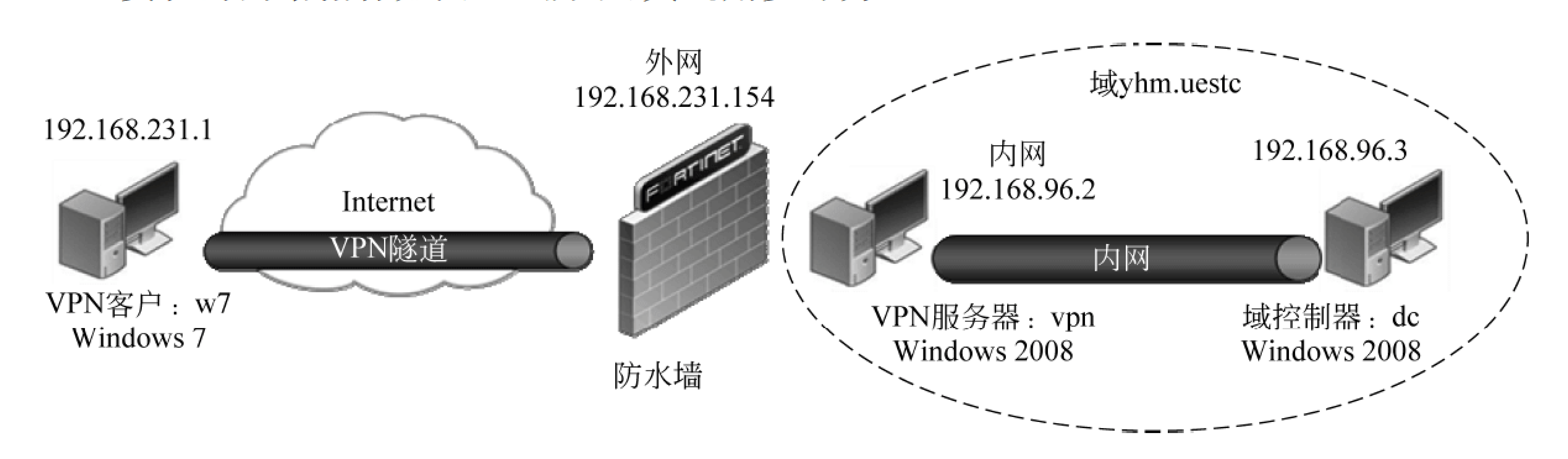


图 8-2　VPN 实验的网络拓扑

(1) 在域 yhm. uestc 中，配置域控制器 dc. yhm. uestc；

(2) 配置 VPN 服务器 vpn. yhm. uestc；

(3) 配置 VPN 客户端 w7。

域控制器和 VPN 服务器为 VMWare 7 上的虚拟机，网络模式为 Host-only，其中内

网 VMnet1：192.168.96.***，外网 VMnet2：192.168.231.***。VPN 客户端为宿主机。

8.3.2　基于 PPTP 的 VPN 实验

PPTP 是在 PPP 协议的基础上开发的一种新的增强型安全协议，支持多协议虚拟专用网(VPN)，可以通过密码身份验证协议(PAP)、可扩展身份验证协议(EAP)等方法增强安全性。可以使远程用户通过拨入 ISP、通过直接连接 Internet 或其他网络安全地访问企业网。基于 PPTP 的 VPN 实验步骤如下：

(1) 本实验中，首先配置域控制器(见图 8-3)，同时还需设置远程用户的具有拨入的权限(见图 8-4)。

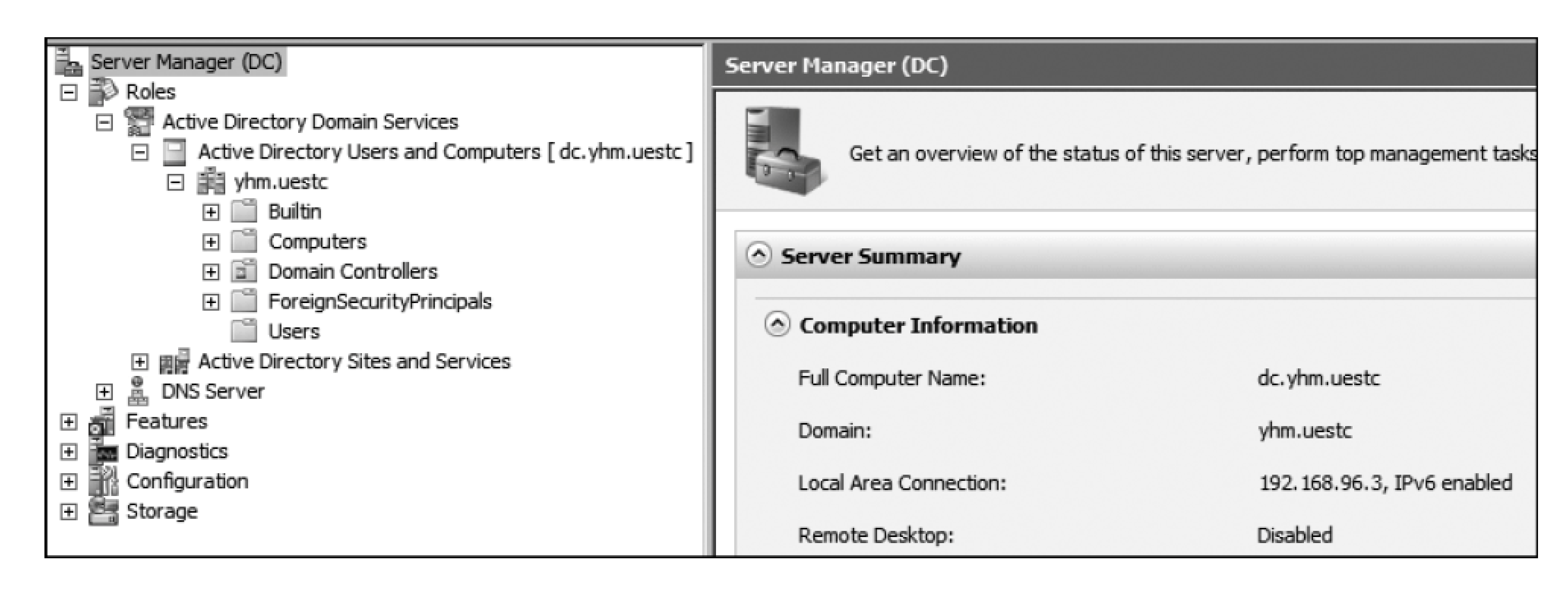


图 8-3　域控制器配置

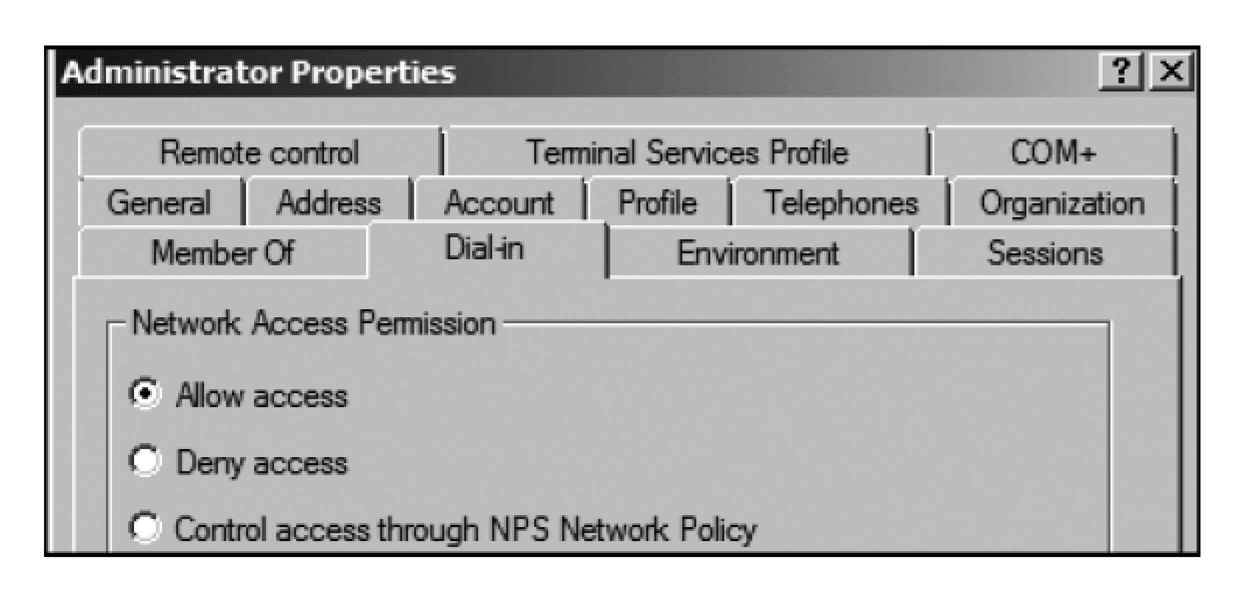


图 8-4　Administrator 允许拨入

(2) 配置 VPN 服务器和“路由和远程访问服务”(见图 8-5)，其中认证和记账都由 Windows 自身提供，不使用 Radius 服务器(见图 8-6)。地址分配使用静态地址池，不使

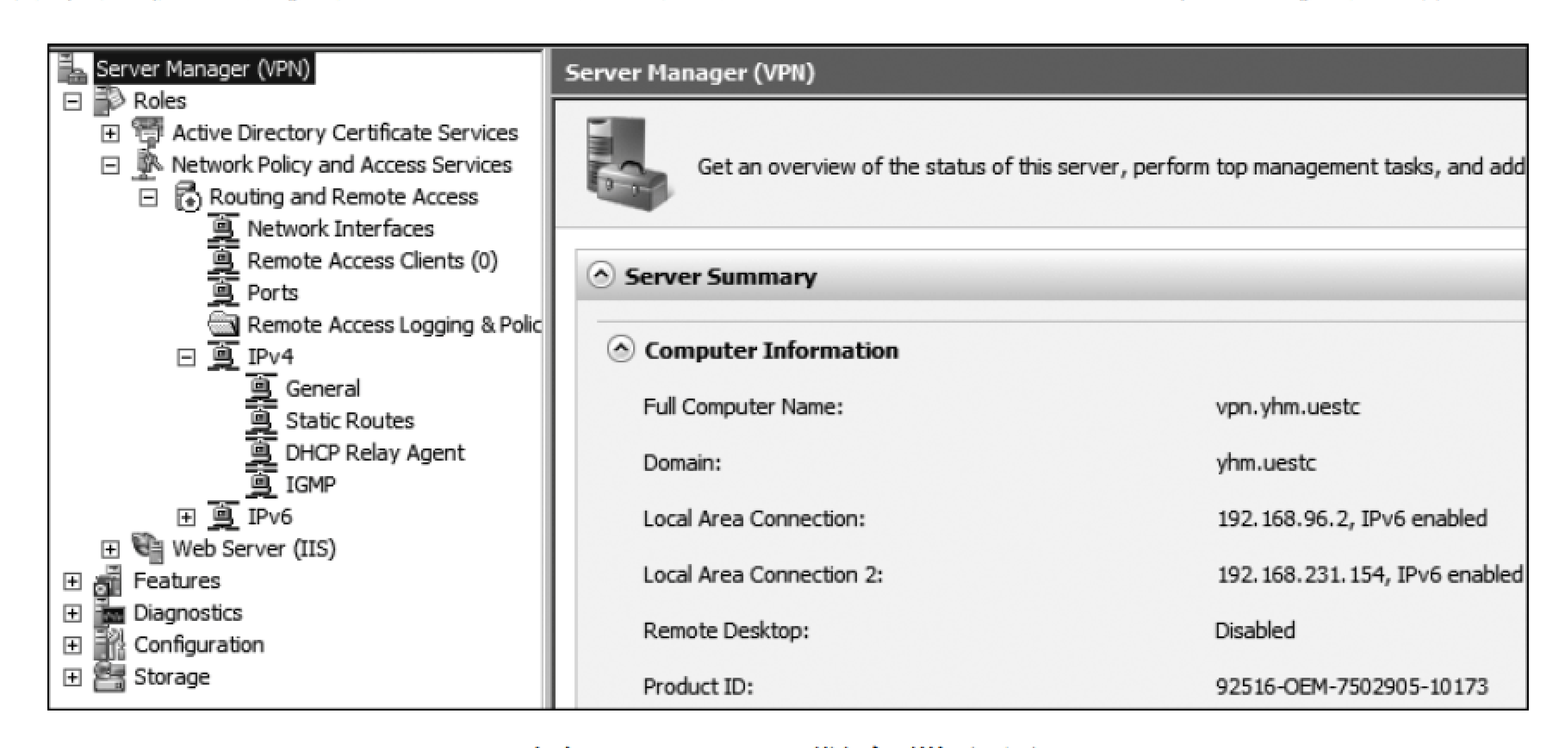


图 8-5　VPN 服务器配置

用 DHCP(见图 8-7)。

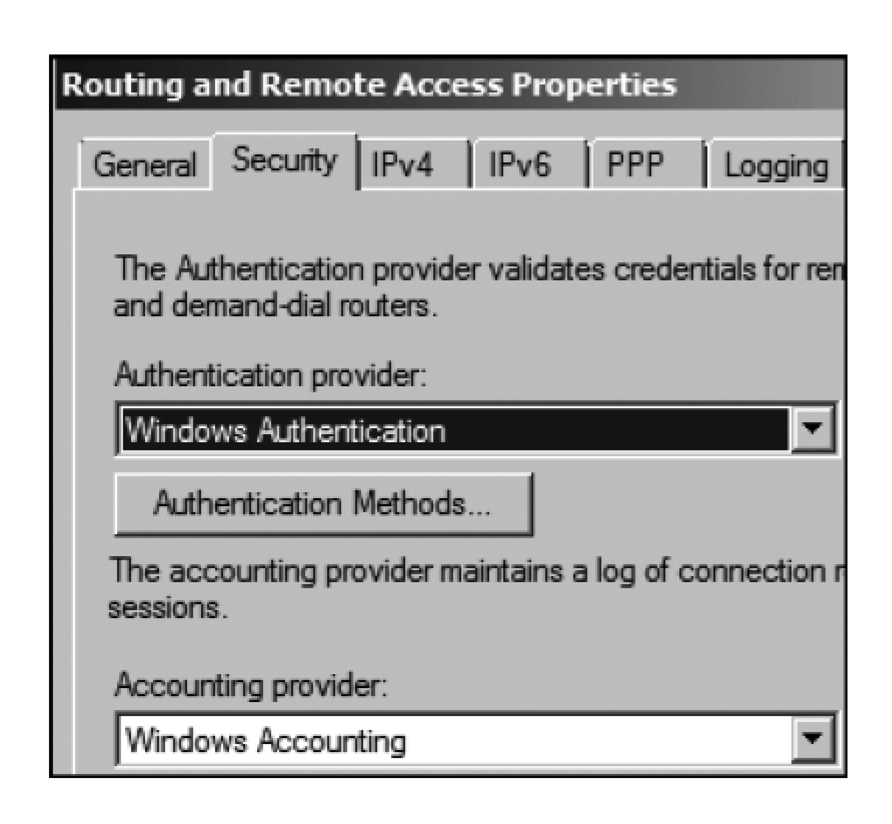


图 8-6　VPN 的认证和记账

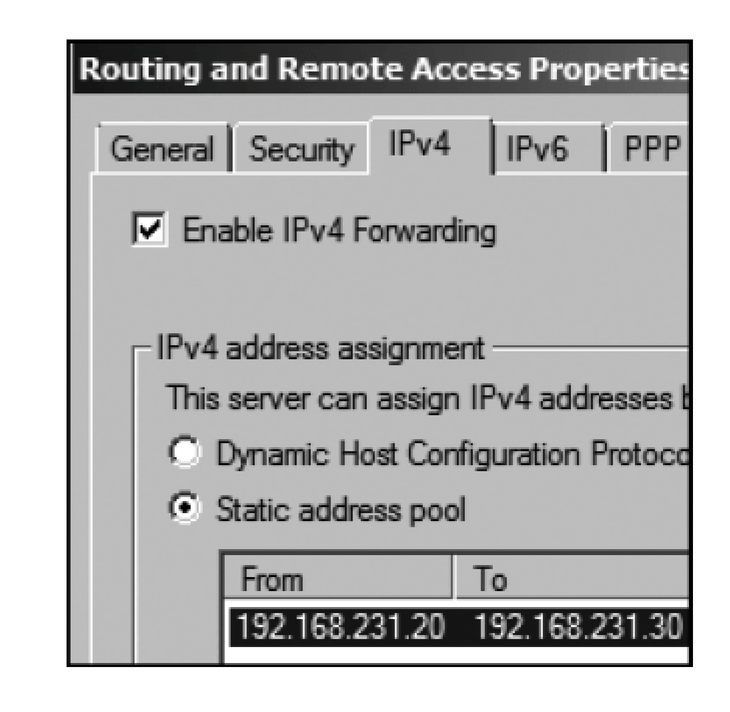


图 8-7　VPN 的地址分配

(3) 在宿主机上配置 VPN 客户端(见图 8-8),其中 VPN 类型选择 PPTP。需要注意的是,在宿主机上,需在"%windir%\system32\drivers\etc\hosts"加入一条:"192.168.231.154　　vpn.yhm.uestc"以解析 vpn.yhm.uestc 这个域名。



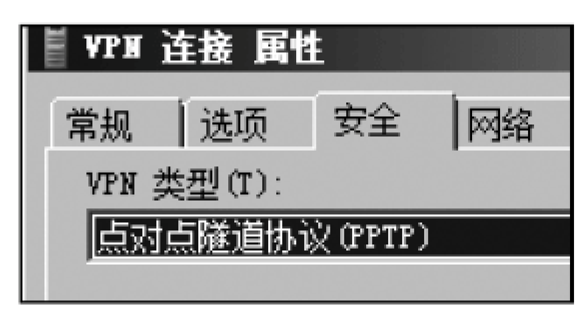


图 8-8　VPN 客户端的配置—PPTP

(4) 进行 VPN 连接测试,测试结果见图 8-9。

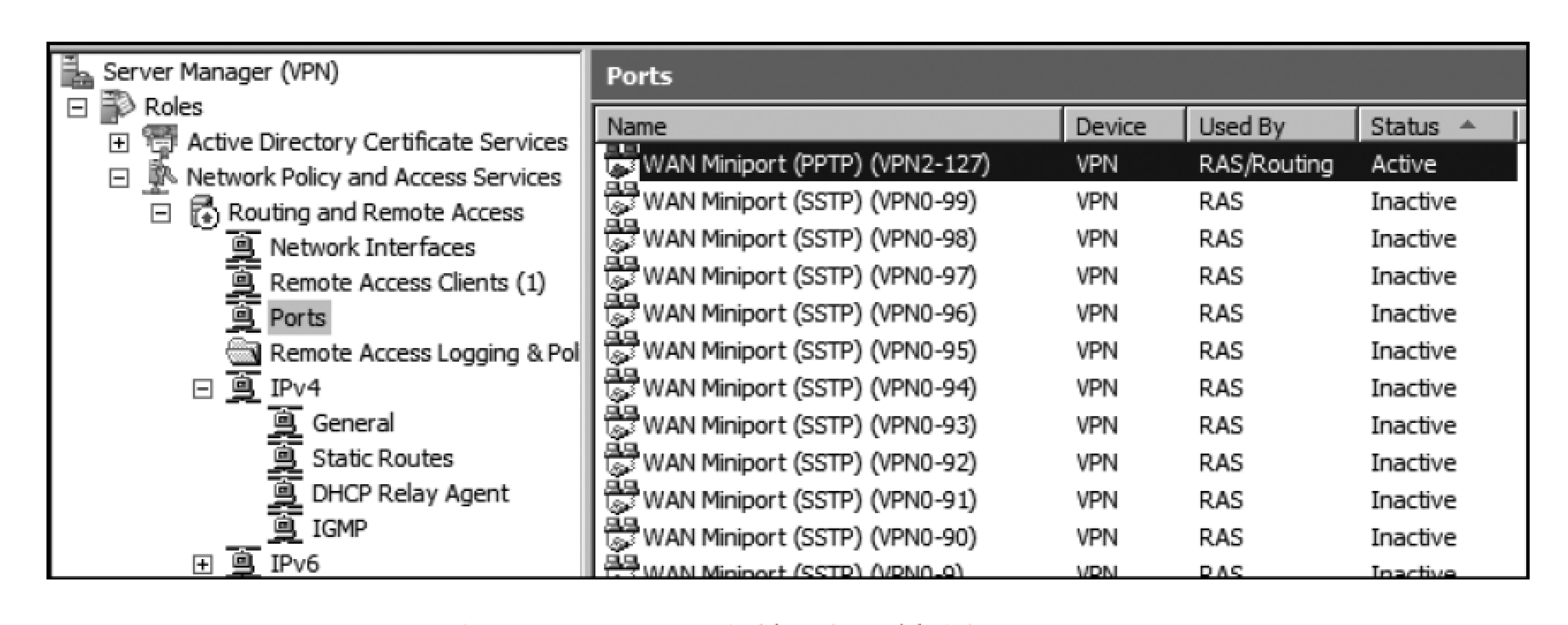


图 8-9　VPN 连接测试结果—PPTP

8.3.3　基于 L2TP over IPSec 的 VPN 实验

L2TP (Layer Two Tunneling Protocol,第二层通道协议)是 VPDN(虚拟专用拨号网络)技术的一种,专门用来进行第二层数据的通道传送,即将第二层数据单元,如点到点协议(PPP)数据单元,封装在 IP 或 UDP 载荷内,以顺利通过包交换网络(如 Internet),抵达目的地。IPSec 协议通过相应的隧道技术,可实现 VPN。

本实验设置 IPSec 策略为预先共享的密钥(见图 8-10 和图 8-11),实验结果见图 8-12。

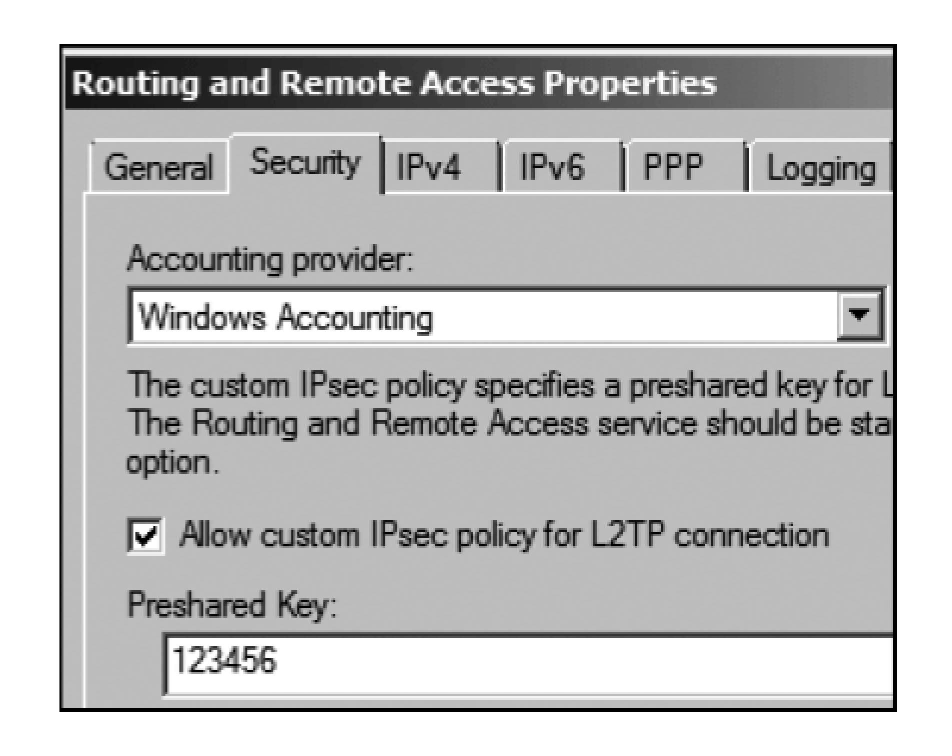


图 8-10 VPN 服务器 L2TP 设置

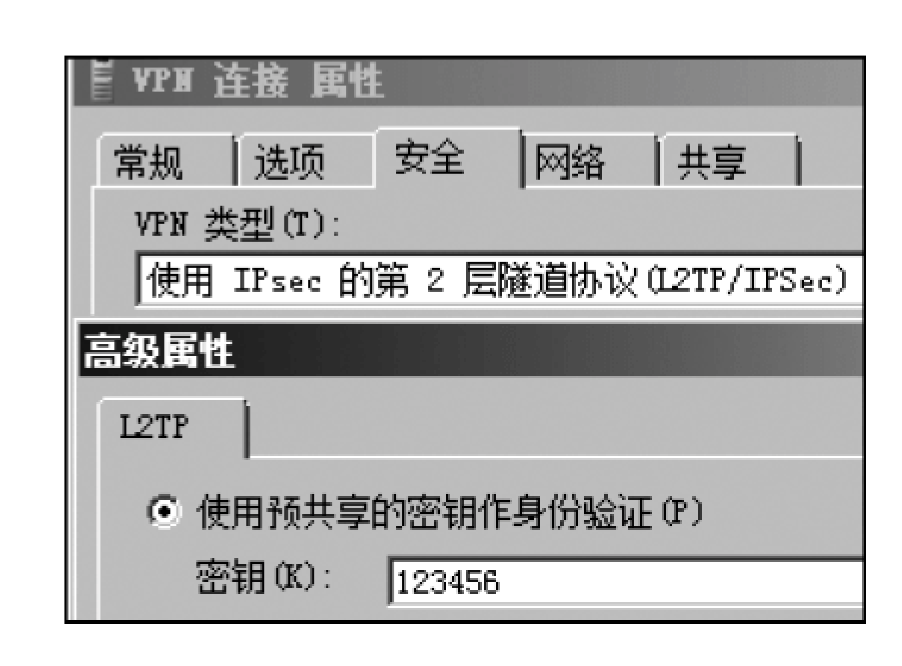


图 8-11 VPN 客户端 L2TP 设置

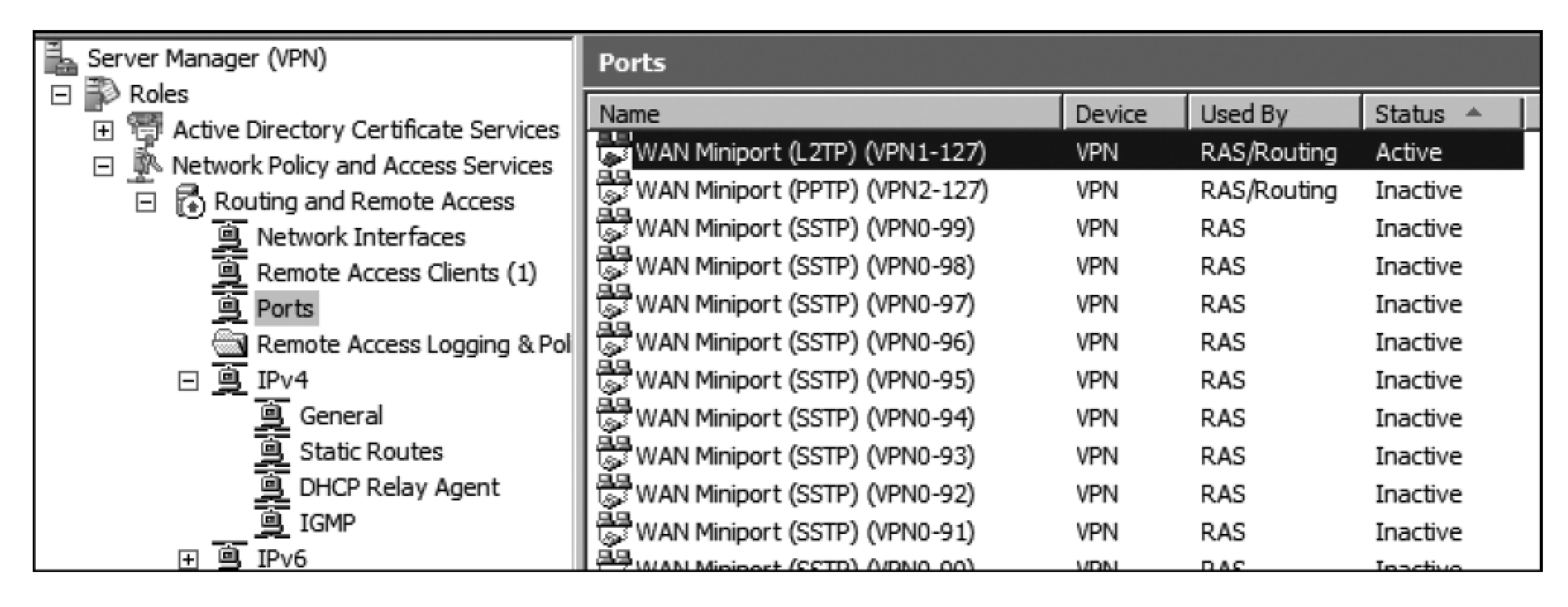


图 8-12 VPN 连接测试结果—L2TP

8.3.4 基于 SSTP 的 VPN 实验

SSTP 提供了一种机制，通过 HTTPS(SSL)建立 VPN 隧道，使用 TCP 端口 443 侦听。同时还使用 PPP，支持强大的认证方法，如 IEEE 802.11i 中的 EAP-TLS 认证。安全套接字层(SSL)提供了增强的传输安全、加密和完整性检查。

本实验中，首先在 VPN 服务器上配置认证中心 CA：yhm-VPN-CA，并为 VPN 服务器申请计算机证书(见图 8-13)。同时，在 VPN 客户端安装根证书并设置 VPN 类型为 SSTP(见图 8-14)。实验结果见图 8-15。

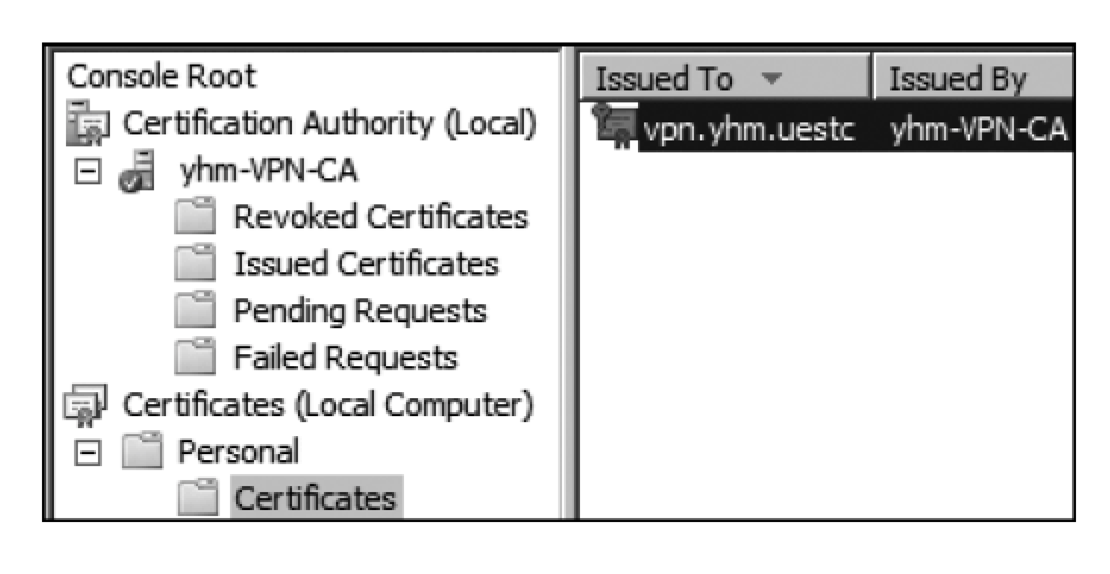


图 8-13 VPN 服务器证书

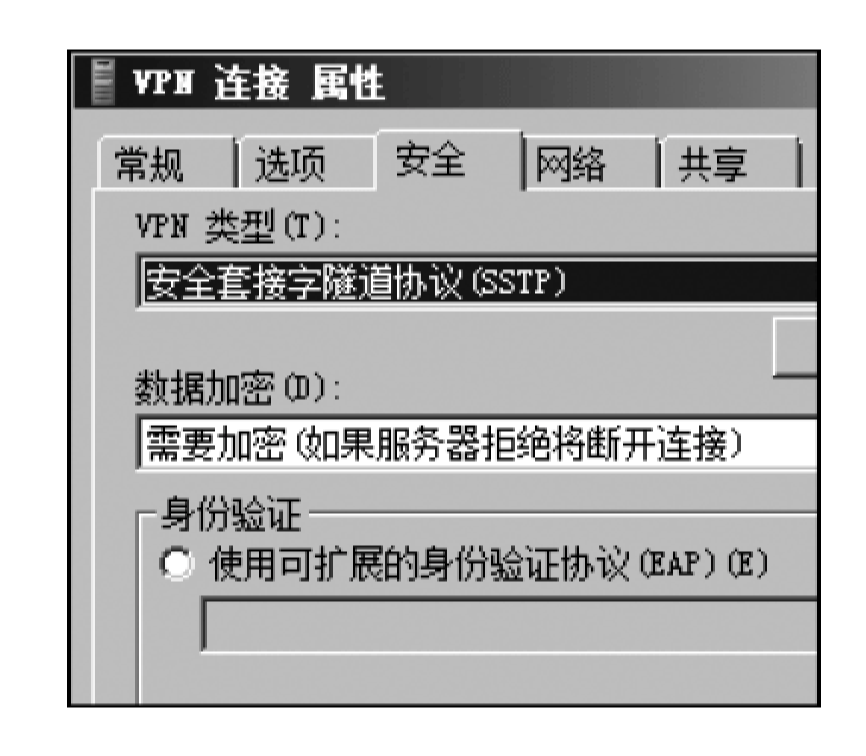


图 8-14 VPN 客户端 SSTP 设置

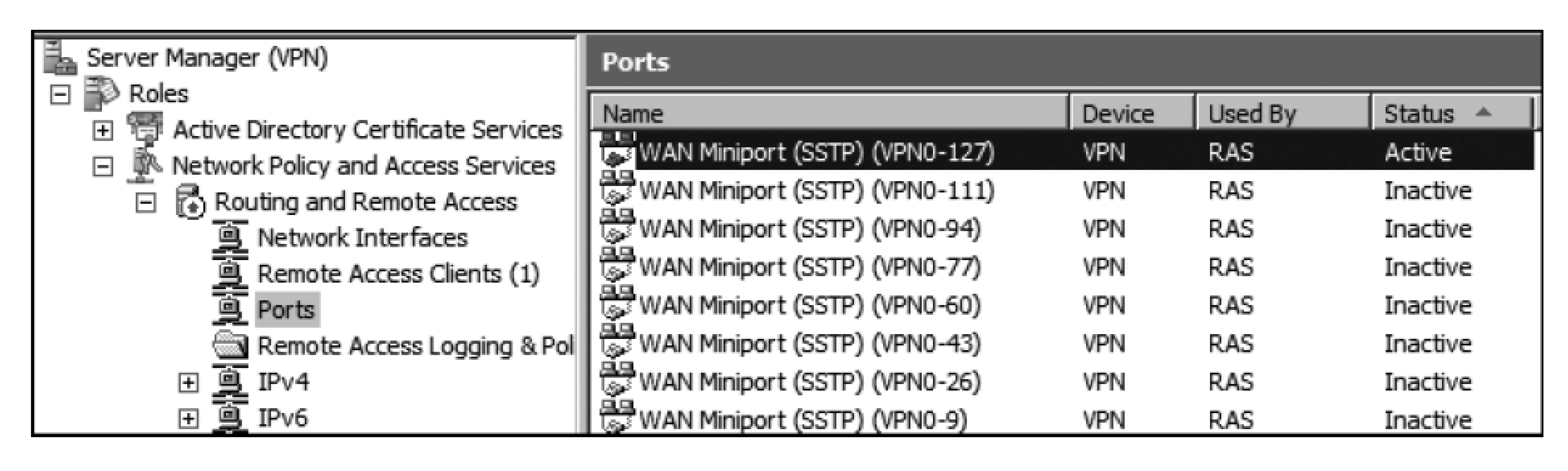


图 8-15　VPN 连接测试结果—SSTP

8.4 小结

本章设计了 VPN 综合实验，它是在 Windows Server 2008 平台下，分别基于 PPTP、L2TP 和 SSTP 三种隧道来进行。一般的网络安全实验教材所设计的 VPN 实验通常是基于简单的 PPTP 协议；本章设计了“基于 L2TP/IPSec”和“基于 SSTP”的 VPN，填补了在本科网络安全实验设计上的一个空白。本实验涉及隧道技术、加解密技术、身份验证技术等多种网络安全技术，这将有助于读者综合掌握网络安全协议的基础知识，以及更好地培养读者的实践工程能力。

参 考 文 献

[1]　王淑江. Windows Server 2012 活动目录管理实践[M]. 北京：人民邮电出版社，2014.

[2]　高晓飞. 网络服务器配置与管理：Windows Server 2003 平台[M]. 北京：高等教育出版社，2009.

[3]　高升. Windows Server 2003 系统管理(第 3 版)[M]. 北京：清华大学出版社，2010.

[4]　S. Wainner. IPSec VPN 设计[M]. 袁国忠译. 北京：人民邮电出版社，2012.

[5]　寻大勇. SSL VPN 网络安全技术的应用研究[J]. 通信技术，2009(10).

[6]　周敬利，曾海鹏. SSL VPN 服务器关键技术研究[J]. 计算机工程与科学，2005(6).

第9章 基于身份加密算法的综合实验

在一般的32位计算机上，计算机最多支持64位的整数运算。但是在公钥密码体系中，64位的整数是远远不够的。比如说RSA公钥算法，它的安全性主要依赖于大整数的难分解性，现在RSA大多数标准要求使用1024位密钥，不再使用512位密钥。对安全性来说，是随着密钥长度的递增而增强的。所以实现大整数的运算是必不可少的。而且，在公钥加密算法中都要用到生成大素数，大整数加减乘除等运算。可以说，大整数运算是支撑公钥密码体系的基石。大整数库实现的效率直接影响着公钥加解密算法的效率。故能否较高效率地实现大整数的基本运算是十分重要的。现在比较流行的大整数库有很多，比如说MIRACL大整数库、CRYPTO++大整数库及FLINT/C软件包等。

然而传统密码学算法实验，仅仅是针对32位的整数，这远远不能够满足编写实践的密码算法的要求，因此本章将在MIRACL大整数库的基础上，进行密码算法的实验。为此，本章选用了比较新颖的基于身份公钥加密算法来进行实验。特别的是，采用了IETF组织所推荐的一个基于身份公钥加密算法：BF-IBE(http://www.ietf.org/rfc/rfc5409.txt)。

9.1 基于身份加密算法

9.1.1 IBE简介

在传统的公钥密码学中，公钥是与身份无关的随机字符串，存在如何认证公钥的真实性的问题。公钥基础设施(Public Key Infrastructure，PKI)通过使用可信任第三方——签证中心(Certification Authority，CA)颁发公钥证书的形式来绑定公钥和身份信息。不过，PKI证书管理复杂，需要建造复杂的CA，证书发布、吊销、验证和保存需要占用较多资源，这就限制了PKI在实时和低带宽的环境中的应用。

为了简化公钥证书的管理，引入了基于身份加密(Identity-Based Encryption，IBE)。在IBE中，公钥是代表用户身份的任意字符串，例如用户的名字、E-mail地址、手机号码等；存在一个可信任的机构——私钥生成器(Private Key Generator，PKG)，根据用户的身份生成相应的私钥：首先运行Setup算法，生成系统的全局参数(或者称为主公钥)和主密钥；然后运行Extract算法，输入主密钥和一个任意的身份ID$\in\{0,1\}^*$，输出相应的私钥。由于公钥直接从身份信息中提取，则证书和公钥目录是不必要的，因此简化了公钥

的管理，并由此带来了不需要密钥信道的非交互式通信以及不需要证书校验，节约了计算和通信成本。

IBE 通常包含了下面四个算法：

- 系统建立（Setup）——生成公开系统参数和主密钥；
- 私钥生成（Extract）——由用户身份和主密钥导出私钥；
- 加密（Encryption）——由用户身份加密；
- 解密（Decryption）——由用户私钥解密。

9.1.2 BF-IBE 方案

由 Boneh 和 Franklin 提出的 IBE 方案中，发送方 Alice 用接受方 Bob 的身份信息 ID 加密消息。接受方 Bob 向（扮演可信任第三方角色）私钥产生中心（PKG）证实身份以获得与自己身份相应的私钥，这样 Bob 就可以对收到的消息进行解密。

假定$(G_1,+)$，$(G_2,\cdot)$为素数 q 阶 GDH 群，双线性对 $e:G_1\times G_1\rightarrow G_2$。$ID_{Bob}$ 代表 Bob 有效的身份信息。基于身份的加密方案需要完成以下几个过程：

1. 系统建立

（1）PKG 产生两个素数 q 阶 GDH 群$(G_1,+)$，$(G_2,\cdot)$和双线性对 $e:G_1\times G_1\rightarrow G_2$。任意选取生成元 $P\in G_1$。

（2）PKG 随机挑选主密钥 $S_{PKG}\in Z_P^*$，计算公钥 $P_{pub}=S_{PKG}P$。

（3）选择一个强哈希函数 $H_1:\{0,1\}^*\rightarrow G_1$，把用户的 ID 映射成 G_1 中的元素。

（4）选择一个强哈希函数 $H_2:G_2^*\rightarrow\{0,1\}^N$，用于确定明文消息的字节长度，$N$ 表示明文消息的字节长度。

（5）PKG 保留 S_{PKG}作为自己的私钥，公开系统参数

$$\mathrm{PARAMS}=(G_1,G_2,e,P,P_{pub},H_1,H_2)$$

2. Bob 私钥提取

（1）Bob 向 PKG 提交各自身份信息 ID_{Bob}，通过 PKG 的身份验证（例如，物理识别和确定 ID 的有效性和唯一性）后，PKG 计算 Bob 的公钥

$$Q_{Bob}=H_1(ID_{Bob})$$

（2）PKG 计算 Bob 的私钥并返回给 Bob。

$$S_{Bob}=S_{PKG}\cdot Q_{Bob}$$

3. Alice 加密

（1）发送方 Alice 首先获得系统参数$(G_1,G_2,e,P,P_{pub},H_1,H_2)$，然后计算

$$Q_{Bob}=H_1(ID_{Bob})$$

（2）Alice 把消息分成 N 位字节的分组 $M\in\{0,1\}^N$，然后随机选一个数 $r\in Z_P^*$，计算密文 C

$$C=(rP,M\oplus H_2(e(Q_{Bob},rP_{pub})))$$

4. Bob 解密

(1) Alice 收到密文 C 后，使用自己的私钥 S_{Bob} 计算

$$H_2(e(S_{Bob},rP))$$

(2) Alice 然后计算

$$H_2(e(S_{Bod},rP))\oplus M\oplus H_2(e(Q_{Bob},rP_{pub}))=M$$

9.2 MIRACL 软件包的实验

9.2.1 软件包简介

MIRACL(Multi-precision Integer and Rational data-type Arithmetic C language Library)是一个完全开源的多精度整数和浮点数算术 C/C++ 库，包括超过 100 个例程，覆盖了多精度算术和密码学(例如 DES、AES、SHA、Diffie-Hellman、ECC、RSA、DSS、IBE 等)的各个方面。它定义了两种大数类型：big(大整数)和 flash(大浮点数)。所有的例程都为速度和效率做了完全的优化，特别是针对那些对执行效率有重要影响的例程，提供绝大多数特定于处理器/编译器的汇编语言的实现，但同时仍然提供了 ANSI C 版本。MIRACL 库还提供了 C++ 接口。

MIRACL 库已经成功地安装在 VAX11/780、各种 UNIX 工作站(Sun、SPARC、IBM RS/6000)、IBM PC(Microsoft C 和 C++ 编译器、Borland Turbo C 和 Borland C++ 编译器、Watcom C 编译器、DJGPP GNU 编译器)、ARM、Apple Macintosh，也可运行在 Itanium 和 AMD 64-bit 处理器。

9.2.2 大整数在 C 语言中的表示(以 FLINT/C 软件包为例)

在公钥密码体制中，参加运算的是长达几百位的大整数。FLINT/C 软件包定义了特殊的数据结构表示大整数。

(1) 出于内存管理会消耗计算时间的考虑，所有的大整数都是静态存储的。

(2) 出于效率原因，要使大整数的运算直接化为 CPU 寄存器的运算，因此，大整数的分量采用标准数据类型。FLINT/C 选择了 unsigned short int(下面称为 USHORT)。

(3) 一个大整数在内存中表示格式为

$$n=(n_l\cdots n_2n_1)_B,\quad 0\leqslant n_i\leqslant B$$

其中，B 表示大整数的基，FLINT/C 软件包中，$B=2^{16}=65536$，l 表示以 B 为基的大整数的位数，即大整数含有多少个 USHORT，$0\leqslant l\leqslant$CLINTMAXDIGIT，若 $n=0$ 时，$l=0$。CLINTMAXDIGIT 表示大整数以 B 为基的最大位数，FLINT/C 软件包定义为 256，可以根据需要调整。

(4) 对应的数据结构可以通过

```
typedef unsighed short clint
typedef clint CLINT [CLINTMAXDIGIT+1]
```

来定义,按照此定义,大整数通过 CLINT n_l 来声明。

9.2.3　MIRACL 的安装和配置实验

【实验目的】

(1) 掌握 MIRACL 软件包的典型安装和配置方法;

(2) 理解大整数在 C 语言中的表示;

(3) 了解 MIRACL 软件包的主要功能。

【实验内容】

(1) 编译并运行 config. c(不带任何优化选项),针对目标计算机/编译器和所希望的功能,作出适当的配置。运行的结果为 mirdef. tst 和 miracl. lst 文件,重命名 mirdef. tst 为 mirdef. h,mirdef. h 为编译 MIRACL 库最适合的头文件。miracl. lst 表明了 MIRACL 库所希望包含的模块。

(2) 对于比较耗时的运算,包括 muldiv、muldvm、muldvd、muldvd2 和 imuldiv,为了加快速度,需要用特定处理器或编译器的汇编语言来实现。mrmuldv. any 包含绝大部分处理器或编译器的汇编语言实现。从 mrmuldv. any 中提取合适本处理器和编译器的 mrmuldv. c 文件,如果不是特别考虑运算速度,也可直接复制标准 C 版本的 mrmuldv. ccc 文件为 mrmuldv. c。

(3) 如果选择了快速的模乘方法 KCM 或 Comba,在任何工作站上编译并运行 mex. c,自动产生 mrcomba. c 或 mrkcm. c。这要求指定处理器或编译器的文件 xxx. mcs,并且编译器支持内联汇编。

(4) 确定编译器所需的所有头文件是可访问的,包括 miracl. h、mirdef. h 以及 ANSI C 常用的头文件 stdlib. h 等,编译 miracl. lst 文件中所包含的文件。

(5) 创建 MIRACL 库 miracl. lib。

【实验环境】

安装 Windows 操作系统的 PC 1 台,其上安装 VC++ 6.0 以上版本的编译器。

【实验参考步骤】

以 Microsoft C++ 6.0 为例,注意"Release" build 比"Debug" build 更快。

(1) 编译和运行 config. c(不带任何优化选项),重命名 mirdef. tst 为 mirdef. h。注意 Microsoft C 的 64-bit 整数类型为__int64。

(2) 新建 Win32 Static Library 类型项目,然后单击 Finish 按钮。

(3) 添加 miracl. lst 中的所有的 mr * . c 到项目。

(4) 单击 Project 选项,选中 Settings 中的 C/C ++ 页,选择 preprocessor,在 Additional Include Directories 中指定 miracl 头文件(miracl. h 和 mirdef. h)的路径。

(5) build 项目创建 MIRACL 库 miracl. lib。

【实验报告】

(1) MIRACL 软件包基于 80x86/VC++ 的安装和配置过程;

(2) MIRACL 或者 FLINT/C 软件包中大整数的数据结构。

9.3 基于身份加密算法的实验

9.3.1 系统参数生成实验

【实验目的】

(1) 掌握基于身份加密方案中的系统参数生成算法;

(2) 了解椭圆曲线密码体制(ECC)的主要思想。

【实验内容】

(1) 创建一个空的 Win32 Console Application 类型的项目 IBE_SET。

(2) 在项目中加入 MIRACL 大数库文件 miracl. lib。

(3) 加入项目主文件 ibe_set. cpp(实现了系统参数生成)。

(4) 加入 zzn2. cpp(实现了有限域 F_{p2})、big. cpp(实现了多精度整数)、zzn. cpp(实现了有限域 F_p)、ecn. cpp(实现了 F_p 上的椭圆曲线)等项目所必需的文件。

(5) 指定路径存放 miracl 头文件: mirdef. h、miracl. h、zzn2. h、big. h、zzn. h、ecn. h。

(6) 编译运行该项目。

(7) 观察运行的结果,公共参数文件 common. ibe 和主密钥文件 master. ibe。

(8) 分析 ibe_set. cpp 源代码,了解实验所需要的特殊椭圆曲线 E/F_p: $y^2=x^3+1$ 是如何产生的。

【实验环境】

安装 Windows 操作系统的 PC 1 台,其上安装 VC++ 6.0 以上版本的编译器。

【实验参考步骤】

以 Microsoft C++ 6.0 为例,注意"Release" build 比"Debug" build 速度更快。

(1) 选择 New→Project 命令,选择 Win32 Console Application 类型,取名为 IBE_SET,单击 Finish 按钮。

(2) 选择 Project→Add to Project→Files 命令,从 Files of type 下拉列表中选择 library files(. lib)选项,将预编译成功的 miracl. lib 加入项目。

(3) 加入项目主文件 ibe_set. cpp。

(4) 加入项目所必需的其他文件 zzn2. cpp、big. cpp、zzn. cpp、ecn. cpp。

(5) 选择 Project,选择 Settings 中的 C/C++ 按钮,选择 Preprocessor,在 Additional Include Directories 中指定头文件路径。

(6) 选择 Build All→Run 命令,编译运行该项目。

(7) 输入 9 位随机数(见图 9-1)。

(8) 观察运行的结果(见图 9-2),以及公共参数文件 common. ibe 和主密钥文件 master. ibe。

【实验报告】

(1) 给出基于身份加密方案中的系统参数生成算法的流程图;

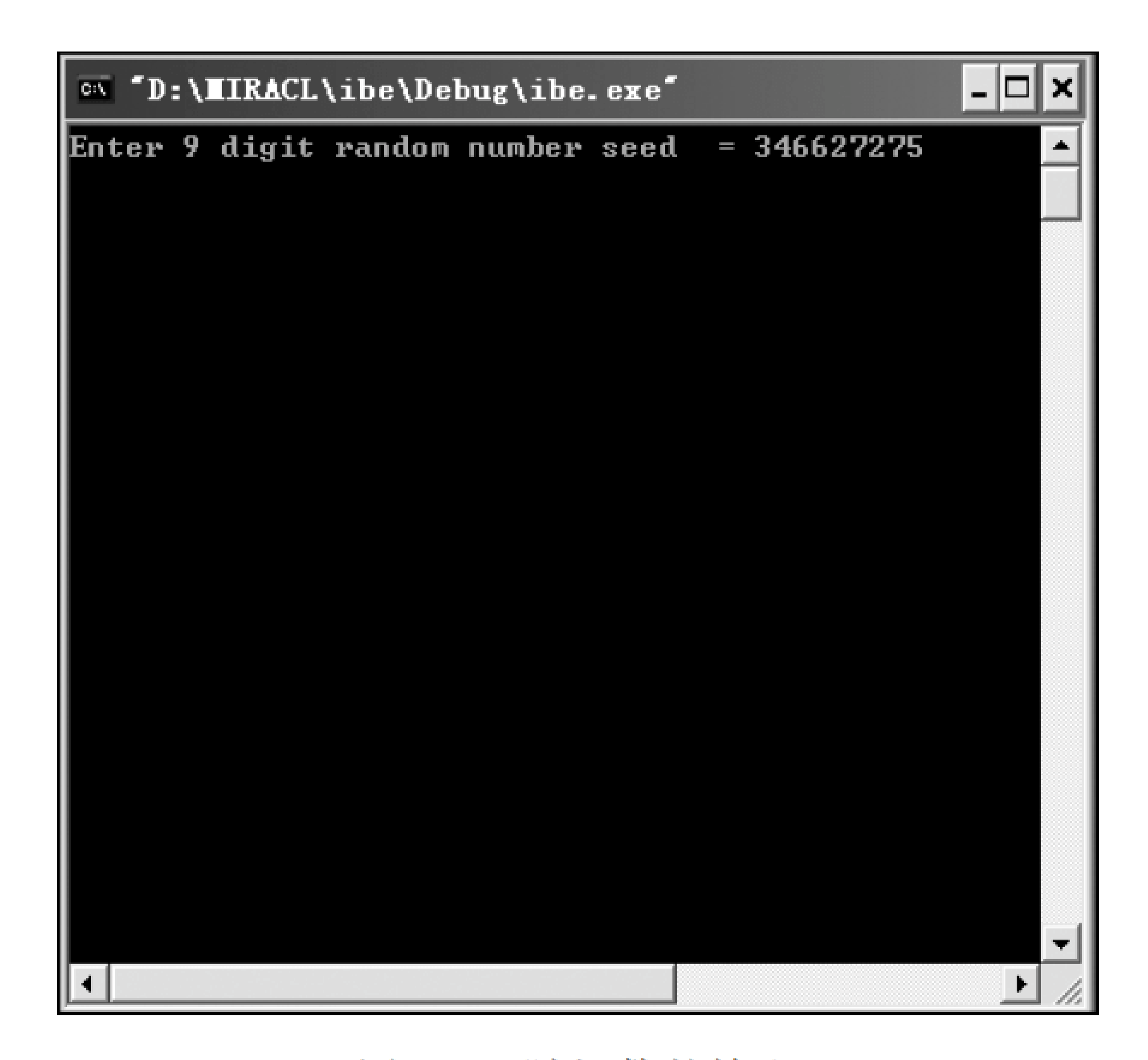


图 9-1　随机数的输入

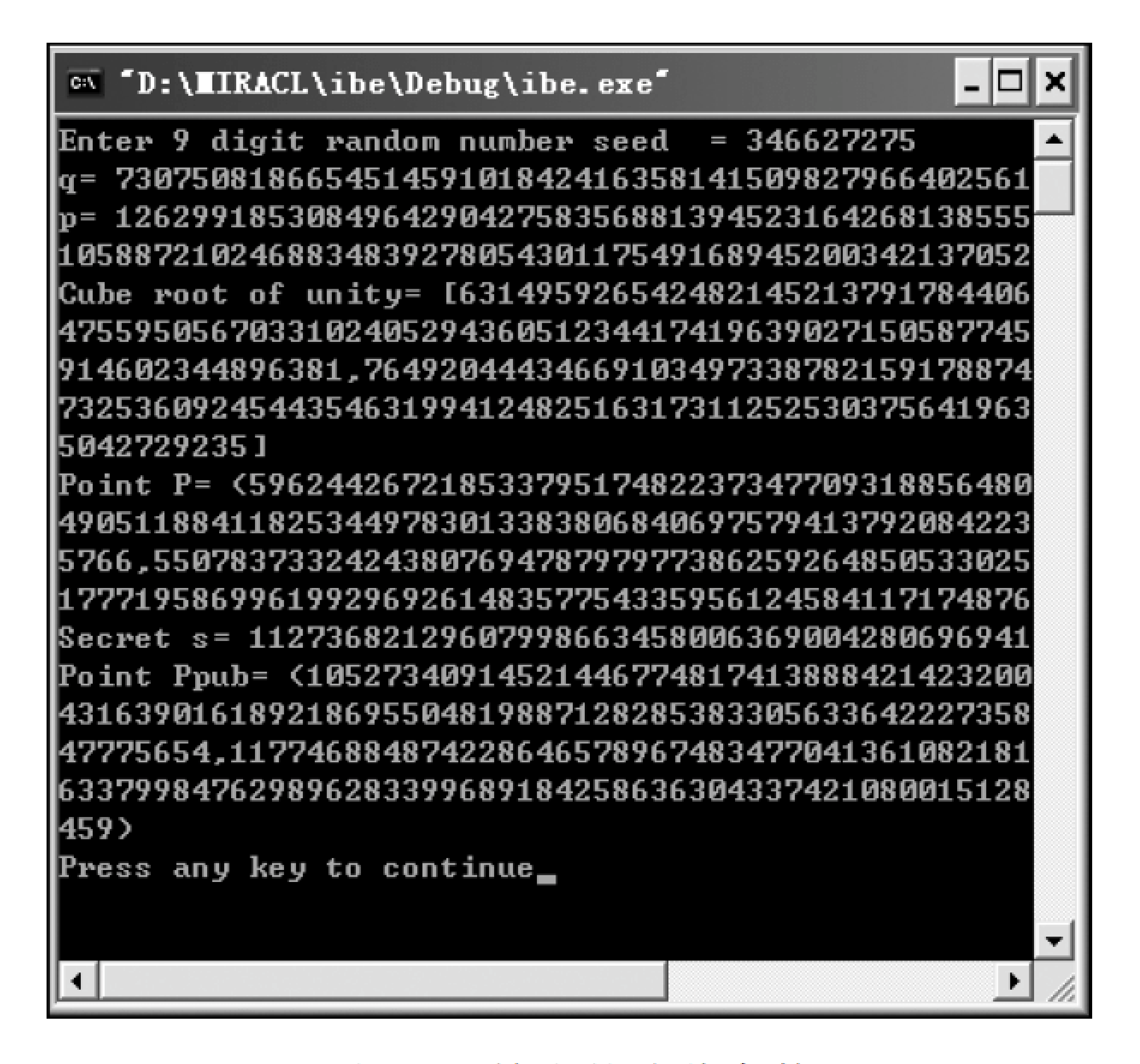


图 9-2　输出的公共参数

(2) 说明所生成的 common.ibe 和 master.ibe 中每一项的含义；

(3) 说明基于身份加密和传统的公钥加密相比的优势。

9.3.2　私钥提取实验

【实验目的】

(1) 掌握基于身份加密方案中的私钥提取算法；

(2) 了解身份是如何 HASH 到椭圆曲线上的点。

【实验内容】

(1) 创建一个空的 Win32 Console Application 类型的项目 IBE_EXT。

(2) 在项目中加入 MIRACL 大数库文件 miracl. lib。

(3) 加入项目主文件 ibe_ext. cpp(实现了私钥提取)。

(4) 加入 big. cpp(实现了多精度整数)、zzn. cpp(实现了有限域 F_p)、ecn. cpp(实现了 F_p 上的椭圆曲线)等项目所必需的文件。

(5) 指定路径存放 miracl 头文件:mirdef. h、miracl. h、big. h、zzn. h、ecn. h。

(6) 编译运行该项目。

(7) 观察运行的结果,存储私钥的文件 private. ibe。

(8) 分析 ibe_ext. cpp 源代码,了解 map_to_point 函数是如何分为两步将身份 HASH 到椭圆曲线上的点。

【实验环境】

安装 Windows 操作系统的 PC 1 台,其上安装 VC++ 6.0 以上版本的编译器。

【实验参考步骤】

以 Microsoft C++ 6.0 为例,注意"Release" build 比"Debug" build 更快。

(1) 选择 New→Project 命令,选择 Win32 Console Application 类型,取名为 IBE_EXT,单击 Finish 按钮。

(2) 选择 Project→Add to Project→Files 命令,从 Files of type 下拉列表中选择 library files (.lib)选项,将预编译成功的 miracl. lib 加入项目。

(3) 加入项目主文件 ibe_ext. cpp。

(4) 加入项目所必需的其他文件 big. cpp、zzn. cpp、ecn. cpp。

(5) 选择 Project,选中 Settings 中的 C/C++ 按钮,选择 Preprocessor,在 Additional Include Directories 中指定头文件路径。

(6) 选择 Build All→Run 命令,编译运行该项目。

(7) 输入身份公钥,这里的身份是 E-mail 地址(见图 9-3)。

(8) 观察运行的结果(见图 9-4),以及私钥文件 private. ibe。



图 9-3　身份公钥的输入

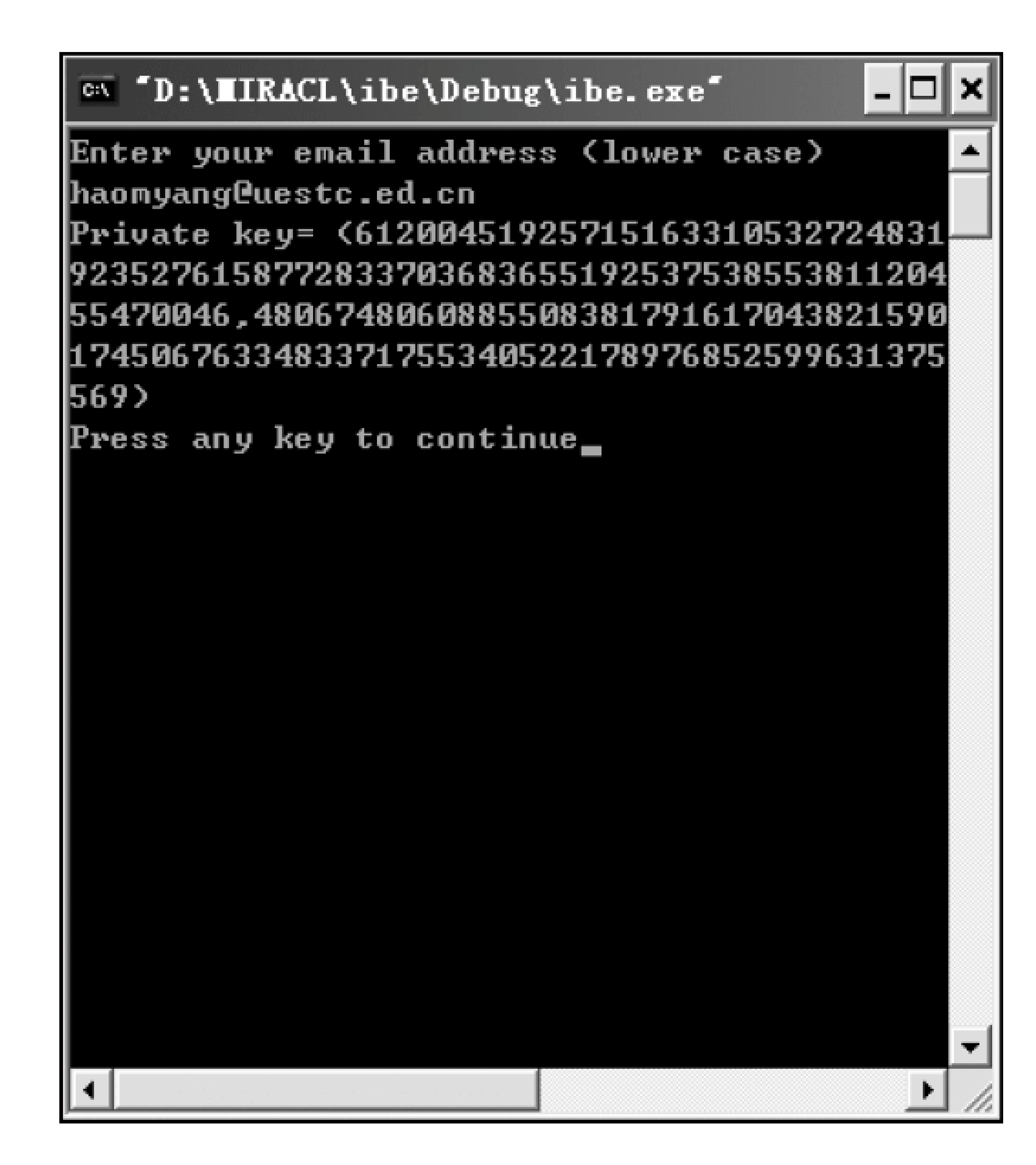


图 9-4　私钥的输出

【实验报告】

(1) 给出基于身份加密方案中的私钥提取算法的流程图；

(2) 说明 map_to_point 函数的运行过程。

9.3.3　加密实验

【实验目的】

(1) 掌握基于身份加密方案中的加密算法；

(2) 了解双线性映射的概念和性质。

【实验内容】

(1) 创建一个空的 Win32 Console Application 类型的项目 IBE_ENC。

(2) 在项目中加入 MIRACL 大数库文件 miracl.lib。

(3) 加入项目主文件 ibe_enc.cpp(实现了加密)。

(4) 加入 zzn2.cpp(实现了有限域 F_{p2})、big.cpp(实现了多精度整数)、zzn.cpp(实现了有限域 F_p)、ecn.cpp(实现了 F_p 上的椭圆曲线)等项目所必需的文件。

(5) 指定路径存放 miracl 头文件：mirdef.h、miracl.h、zzn2.h、ebrick.h、big.h、zzn.h、ecn.h。

(6) 编译运行该项目。

(7) 分析加密的过程：首先产生一个 AES 会话密钥，用它来加密一个明文文件 filename.*，得到密文文件 filename.ibe，再用身份作为公钥来加密 AES 会话密钥，得到 filename.key。

(8) 分析 ibe_enc.cpp 源代码，了解双线性映射和 Tate 对的概念。

【实验环境】

安装 Windows 操作系统的 PC 1 台，其上安装 VC++ 6.0 以上版本的编译器。

【实验参考步骤】

以 Microsoft C++ 6.0 为例，注意"Release" build 比"Debug" build 更快。

(1) 选择 New→Project 命令，选择 Win32 Console Application 类型，取名为 IBE_ENC，单击 Finish 按钮。

(2) 选择 Project→Add to Project→Files 命令，从 Files of type 下拉列表中选择 library files (.lib)选项，将预编译成功的 miracl.lib 加入项目。

(3) 加入项目主文件 ibe_enc.cpp。

(4) 加入项目所必需的其他文件 zzn2.cpp、big.cpp、zzn.cpp、ecn.cpp。

(5) 选择 Project，选择 Settings 中的 C/C++ 页，选择 Preprocessor，在 Additional Include Directories 中指定头文件路径。

(6) 选择 Build All→Run 命令，编译运行该项目。

(7) 创建一个明文文件，例如 plain.txt，在该文件输入一些明文 (见图 9-5)。

(8) 分别输入 9 位的随机数、身份公钥以及明文文件的名字，观察运行的结果(见图 9-6)。

(9) 检查当前运行目录下是否产生了 plain.ibe 和 plain.key 文件。



图 9-5　明文的输入

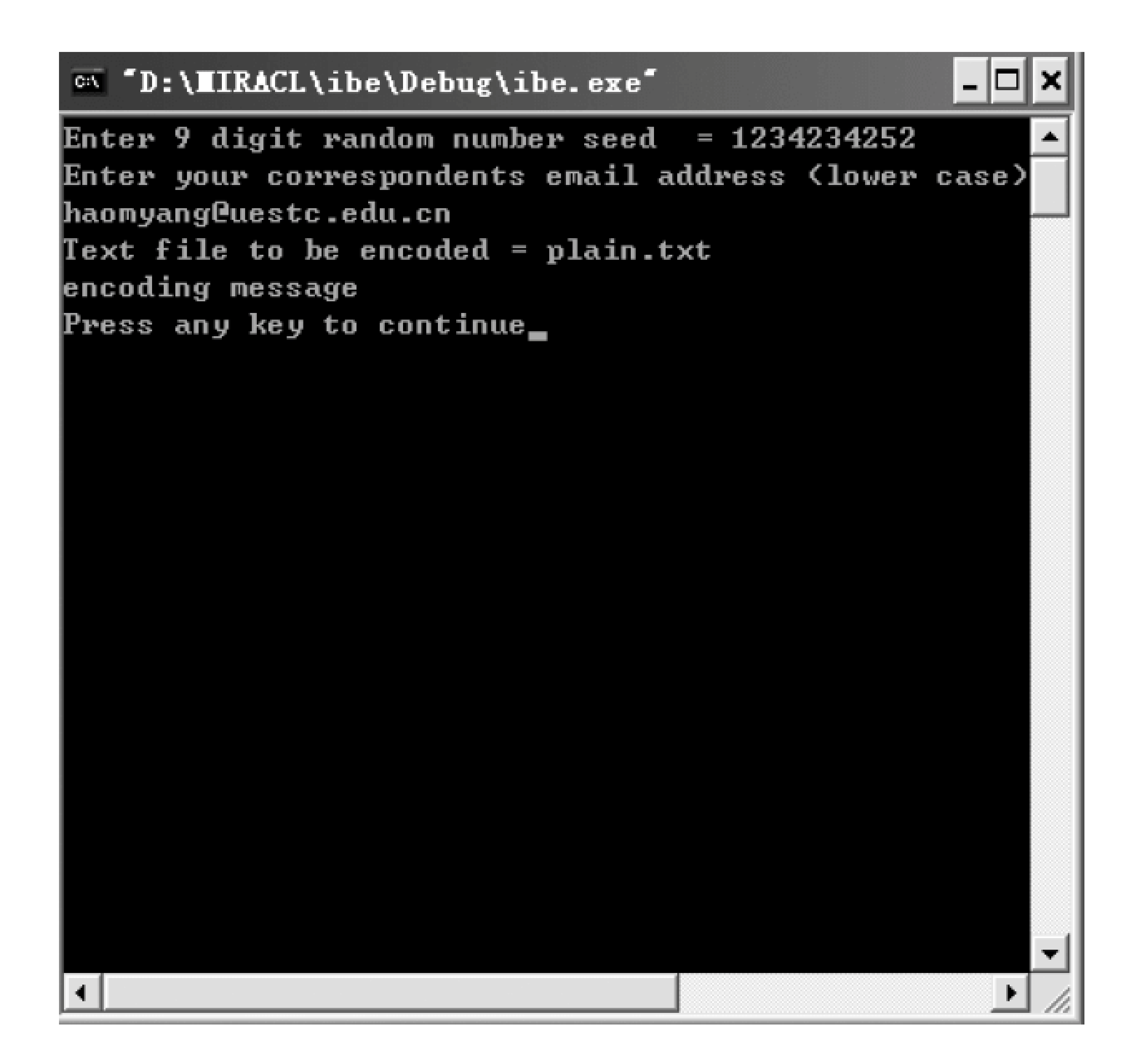


图 9-6　加密的操作

【实验报告】

(1) 给出基于身份加密方案中的加密算法的流程图；

(2) 说明 Tate 对的计算过程。

9.3.4　解密实验

【实验目的】

(1) 掌握基于身份加密方案中的解密算法；

(2) 理解混合加密的思想。

【实验内容】

(1) 创建一个空的 Win32 Console Application 类型的项目 IBE_DEC。

(2) 在项目中加入 MIRACL 大数库文件 miracl.lib。

(3) 加入项目主文件 ibe_dec.cpp(实现了解密)。

(4) 加入 zzn2.cpp(实现了有限域 F_{p2})、big.cpp(实现了多精度整数)、zzn.cpp(实现了有限域 F_p)、ecn.cpp(实现了 F_p 上的椭圆曲线)等项目所必需的文件。

(5) 指定路径存放 miracl 头文件：mirdef.h、miracl.h、zzn2.h、ebrick.h、big.h、zzn.h、ecn.h。

(6) 编译运行该项目。

(7) 分析解密的过程：首先用 private.ibe 中的私钥解密 filename.key，得到 AES 会话密钥，再用 AES 会话密钥解密 filename.ibe，得到明文。

(8) 分析 ibe_enc.cpp 和 ibe_dec.cpp 源代码，了解混合加密的运行流程。

【实验环境】

安装 Windows 操作系统的 PC 1 台，其上安装 VC++ 6.0 以上版本的编译器。

【实验参考步骤】

以 Microsoft C++ 6.0 为例,注意"Release" build 比"Debug" build 更快。

(1) 选择 New→Project 命令,选择 Win32 Console Application 类型,取名为 IBE_DEC,单击 Finish 按钮。

(2) 选择 Project→Add to Project→Files 命令,从 Files of type 下拉列表中选择 library files (.lib)选项,将预编译成功的 miracl.lib 加入项目。

(3) 加入项目主文件 ibe_dec.cpp。

(4) 加入项目所必需的其他文件 zzn2.cpp、big.cpp、zzn.cpp、ecn.cpp。

(5) 选择 Project,选中 Settings 中的 C/C++ 页,选择 Preprocessor,在 Additional Include Directories 中指定头文件路径。

(6) 选择 Build All→Run 命令,编译运行该项目。

(7) 输入需要解密的文件,并观察运行的结果(见图 9-7)。

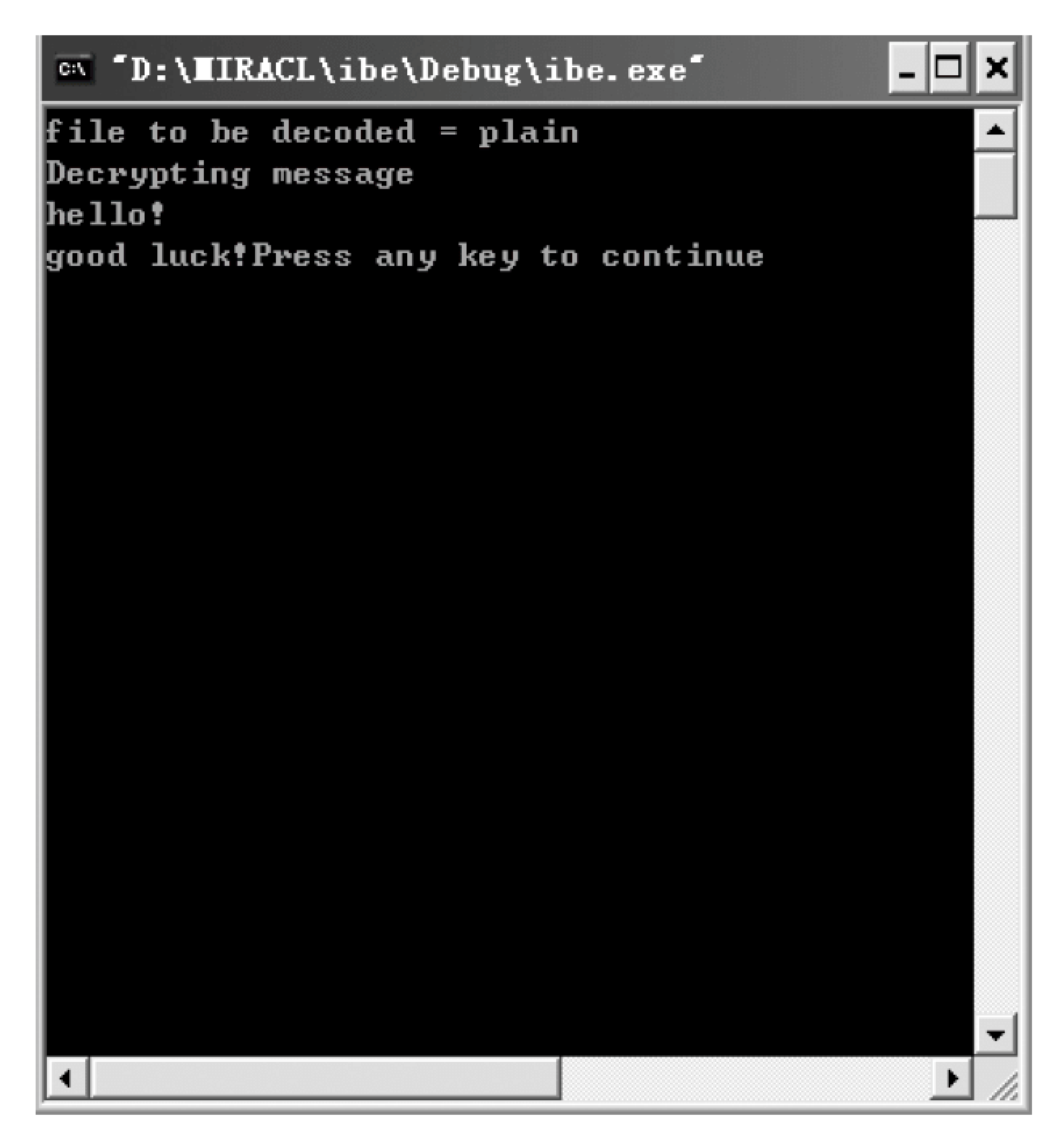
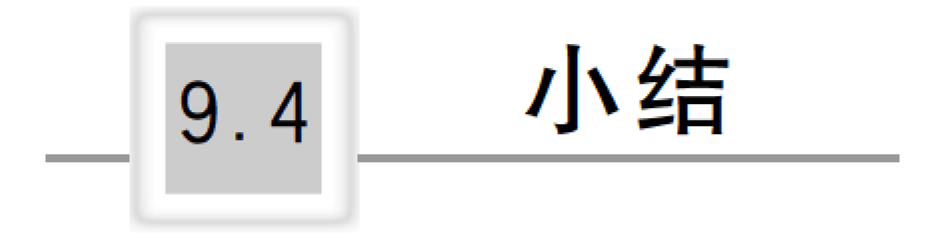


图 9-7 解密的操作

【实验报告】

(1) 给出基于身份加密方案中的解密算法的流程图;

(2) 说明混合加密的优点。



9.4 小结

本章设计了基于身份加密的综合实验,它使用了 MIRACL 大整数的软件包,分别进行了大数软件包 MIRACL 的安装和配置实验、IBE 系统参数生成实验、IBE 私钥提取实验、IBE 加密实验以及 IBE 解密实验。这将有助于读者编写实践的密码算法。

参考文献

[1] 胡亮,赵阔,袁巍等.基于身份的密码学[M].北京:高等教育出版社,2011.

[2] 杨波.现代密码学[M].北京:清华大学出版社,2007.

[3] W. Stallings.密码编码学与网络安全:原理与实践(第四版)[M].北京:电子工业出版社,2006.

[4] A. Kahate.密码学与网络安全(第2版)[M].金名等译.北京:清华大学出版社,2009.

[5] 杨浩淼,邱乐德.基于身份的公钥密码体制的研究[M].成都:电子科技大学出版社,2012.

[6] 曾兵,杨宇,曹云飞.基于身份加密(IBE)技术研究[J].信息安全与通信保密,2011(4).

[7] MIRACL 大数运算库使用手册.

第 10 章 信息探测综合实验

在攻击技术中，首要且不可或缺的一步是探测(Probe)。探测的主要目的便是对主机信息或者网络信息进行收集。通常，探测作为入侵者在发动攻击前一个必需的手段主要采取的技术手段是对目标进行扫描。例如对活动主机、操作系统、开放的端口，以及安全漏洞等关键信息的扫描。

目前，扫描技术有很多种，其中最常用的扫描技术通常包括 Ping 扫描、端口扫描、安全漏洞扫描等。针对这些不同的扫描，利用扫描器来自动检测和获得目标系统的大量信息，例如开放的端口以及提供的一些服务。本实验通过主机信息探测和安全漏洞探测两个方面来介绍两种著名的扫描工具，并以此了解其中的基本原理。

10.1 主机信息探测

主机信息探测主要的目的是检查主机是否在线，如果目标主机在线，将进一步扫描足迹的开放端口，操作系统等信息，为后面的进一步攻击做铺垫。

10.1.1 主机信息探测原理概述

主机信息探测的技术主要包括基于 ICMP 的探测方法和端口扫描。下面针对这两种主要的技术分别做简要阐述。

1. 基于 ICMP 的探测方法

基于 ICMP 的探测方法主要有 ICMP-ECHO 探测、ICMP-SWEEP 探测、广播 ICMP 探测、Non-ECHO ICMP 探测等。下面介绍几种常用的探测方法。

1) ICMP-ECHO 探测

通常可以使用 ICMP-ECHO 数据报来探测主机是否存活，这是最常用也是最简单的一种探测方法。通过简单的发送一个回显(ICMP-ECHO，Type＝8)的 ICMP 数据包，当主机得到请求后，返回一个回显(ICMP ECHO Reply，Type＝0)的数据包，通过是否收到 Ping 的响应包就可以判断主机是否开机。实验原理如图 10-1 所示。

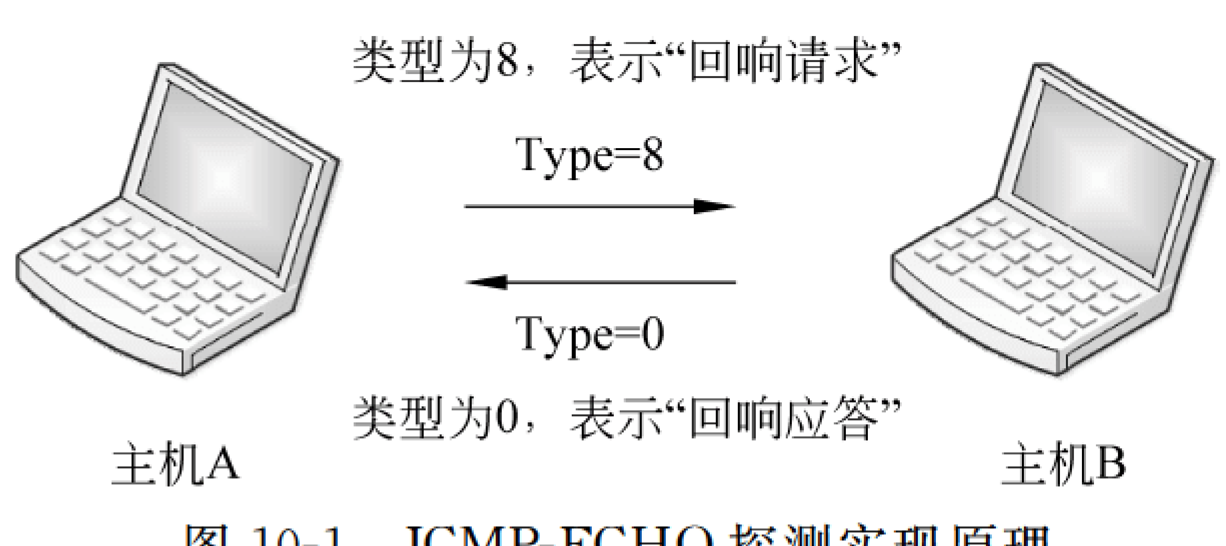


图 10-1　ICMP-ECHO 探测实现原理

2）基于高级 ICMP 的扫描

高级的 ICMP 扫描技术利用了 ICMP 协议的报错功能。当接受端主机发现所接受的数据包协议项出现了错误，那么该主机就返回给源主机一个目标主机不可到达的数据包。需要注意的是，这些数据包当且仅当在错误发生的时候，自动发送给源主机的。

基于 ICMP 探测能够通过 ICMP 数据报来探测端到端的 IP 链路是否工作正常，并且能够及时地判断出故障点，许多网络管理软件都使用 ICMP 来发现网络拓扑。但是其也存在一些不足。例如网络管理员往往会在路由器或者交换机上封锁一些病毒使用的端口，同时过滤掉 ICMP 数据报。

2. 全连接扫描(TCP connect Scan)

端口扫描主要目的是获取目标系统上的端口信息，用于识别其上具有的或者当前运行的 TCP 和 UDP 服务。其中全连接扫描是 TCP 端口扫描的基础，现有的全连接扫描有 TCP connect()扫描和 TCP 反向 ident 扫描等。

全连接扫描是一种最原始，最为容易实现的扫描技术。如果目标主机能够 connect，就说明一个相应的端口打开。全连接扫描主要利用 TCP/IP 协议的三次握手来判断相应的端口是否打开。如果端口开放，则连接将建立成功；否则，若返回 −1 则表示端口关闭。建立连接成功：响应扫描主机的 SYN/ACK 连接请求，这一响应表明目标端口处于监听（打开）的状态。如果目标端口处于关闭状态，则目标主机会向扫描主机发送 RST 的响应。TCP connect()扫描具有实现简单，并且可以用普通用户权限执行不过其也具有一定的不足，例如容易被防火墙检测。

10.1.2　Nmap 工具的使用

1. Nmap 简介

Nmap 是一个网络连接端扫描软件，用来扫描网上计算机开放的网络连接端。确定哪些服务运行在哪些连接端，并且推断计算机运行哪个操作系统（这时亦称 fingerprinting）。它是网络管理员必用的软件之一，以及用于评估网络系统安全。Nmap 可以检测远程主机的下列特征：使用的操作系统、TCP 序列、运行绑定到每个端口上的应用程序的用户名、DNS 名、主机地址是否是欺骗地址，以及其他一些东西。除此之外，Nmap 提供了一些高级的特征，例如，通过 TCP/IP 协议栈特征探测操作系统类型，秘密扫描，动态延时和重传计算，并行扫描，通过并行 Ping 扫描探测关闭的主机、诱饵扫描、避开端口过滤检测、直接 RPC 扫描（无须端口映射）、碎片扫描，以及灵活的目标和端口设定。Nmap 命令的基本语法格式如下：

```
nmap [Scan Type(s)] [Options] <target specification>
```

2. Nmap 重要参数解释

根据上述的语法格式，首先介绍常用的扫描类型选项，即“Scan Type(s)”部分。

(1) -sT

此扫描选项是一种最为基本的 TCP 扫描方式，即 TCP connect()扫描。

TCP connect()扫描：这是最基本的 TCP 扫描方式。扫描主机通过 TCP/IP 协议的三次握手与目标主机建立一次完整的连接。主机的连接由系统的 Connect 开始，如果端口开放则连接将建立成功。这种扫描方法的实现简单，对操作者的权限也没有限制。不过这种扫描很容易被检测到，在目标主机的日志中会记录大批的连接请求以及错误信息。

(2) -sS

此标志代表 TCP 同步扫描(TCP SYN)。SYN 扫描又被称作为半连接扫描。它利用三次握手协议的弱点来实现。扫描器向远程主机端口发出一个 TCP 同步包(SYN)，然后等待回应，如果对方返回 SYN/ACK(响应)包就表示目标端口正在监听；如果返回 RST 数据包，就表示目标端口没有监听程序；如果收到一个 SYN/ACK 包，源主机就会马上发出一个 RST(复位)数据包断开和目标主机的连接，这实际上是由操作系统内核自动完成的。SYN 扫描比 TCP connect 扫描隐秘，主机上不会留下记录，不过需要 root 权限来定制 SYN 数据包。

(3) -sN/-sF/sX

以上三个符号分别表示空(Null)扫描、秘密 FIN 数据包扫描和圣诞树(Xmas Tree)模式。NULL 扫描即发送一个没有任何标志位的 TCP 包，根据 RFC793，如果目标主机的相应端口是关闭的话，应该发送回一个 RST 数据包。FIN 扫描即对某端口发送一个 TCP FIN 数据报给远端主机。如果主机没有任何反馈，那么这个主机是存在的，而且正在监听这个端口；主机反馈一个 TCP RST 回来，说明该主机是存在的，但是没有监听这个端口。sX 扫描与 FIN 扫描的原理类似，不过它将数据报中的 6 个标志位全部设置为 1 后发送给目标主机。若端口开放，目标主机不返回任何信息，若返回 RST 则说明端口关闭。

(4) -sP

Ping 扫描：有时只是想知道此时网络上哪些主机正在运行。通过指定网络内的每个 IP 地址发送 ICMP echo 请求数据包，Nmap 就可以完成这项任务。如果主机正在运行就会作出响应。其实 Nmap 在任何情况下都会进行 Ping 扫描，只有目标主机处于运行状态，才会进行后续的扫描。如果只是想知道目标主机是否运行，而不想进行其他扫描，才会用到这个选项。

(5) -sU

UDP 扫描：这种扫描可以确定主机上提供哪些 UDP 服务。Nmap 首先向目标主机的每个端口发出一个 0 字节的 UDP 包，如果收到端口不可达的 ICMP 消息，端口就是关闭的，否则就假设它是打开的。

(6) -sA

ACK 扫描：发送一个只有 ACK 标志的 TCP 数据报给主机，如果主机反馈一个 TCP RST 数据报来，那么这个主机是存在的。也可以通过这种技术来确定对方防火墙是否是简单的分组过滤，还是一个基于状态的防火墙。

下面介绍可选项部分常用的参数。

(1) -P0

此参数的目的在于设置主机在扫描远程主机之前不必 Ping 主机。这个选项可以对

有些网络的防火墙不允许 ICMP echo 请求穿过。

(2) -PT

此选项设置为在扫描之前,使用 TCP Ping 确定哪些主机正在运行。Nmap 是向目标网络(或者单一主机)发出 TCP ACK 包然后等待回应。如果主机正在运行就会返回 RST 包。只有在目标网络/主机阻塞了 Ping 包,而仍旧允许对其进行扫描时,这个选项才有效。

(3) -PS

此命令需要 root 权限,可以将 Nmap 设置为使用 SYN 包而不是 ACK 包来对目标主机进行扫描。

(4) -PI

这个选项是设置 Nmap 用 ICMP echo 请求来扫描目标主机是否正在运行。不过在使用这个选项让 Nmap 发现正在运行的主机的同时,Nmap 也会对用户的直接子网广播地址进行观察。

(5) -PB

该选项是使用 ACK(-PT)和 ICMP(-PI)两种扫描类型并行扫描。如果防火墙能够过滤其中一种包,使用这种方法,就能够穿过防火墙。

10.1.3 实验 1 Nmap 的使用

【实验目的】

(1) 了解网络扫描技术的基本原理。

(2) 掌握 nmap 工具的使用方法和各项功能。

(3) 通过使用 nmap 工具,对网络中的主机信息等进行探测。

(4) 掌握针对网络扫描技术的防御方法。

【实验内容】

Nmap 工具的使用。

【实验设备与环境】

Nmap 扫描器、PC 一台、目标主机一台。

【实验方法步骤】

(1) 运行 cmd,进入 nmap 所在文件夹,然后在命令行内输入 nmap 命令,显示 Nmap 命令提示符,如图 10-2 所示。

(2) 使用 Nmap 扫描你所在局域网的整个网络寻找目标(此处以 172.22.1.9 网段为例,可根据实验室情况自行设定)。通过使用-sP 命令,进行 ping 扫描,当发现了在目标网络上运行的主机,下一步是进行端口扫描,如图 10-3 所示。现在使用 TCP SYN 扫描,探测 172.22.1.9 的主机信息,其命令是:

```
nmap -sS  172.22.1.9
```

(3) 指定端口扫描:输入命令行 nmap -sS -p 21,23,53,80 -v 172.22.1.9,如图 10-4 所示。

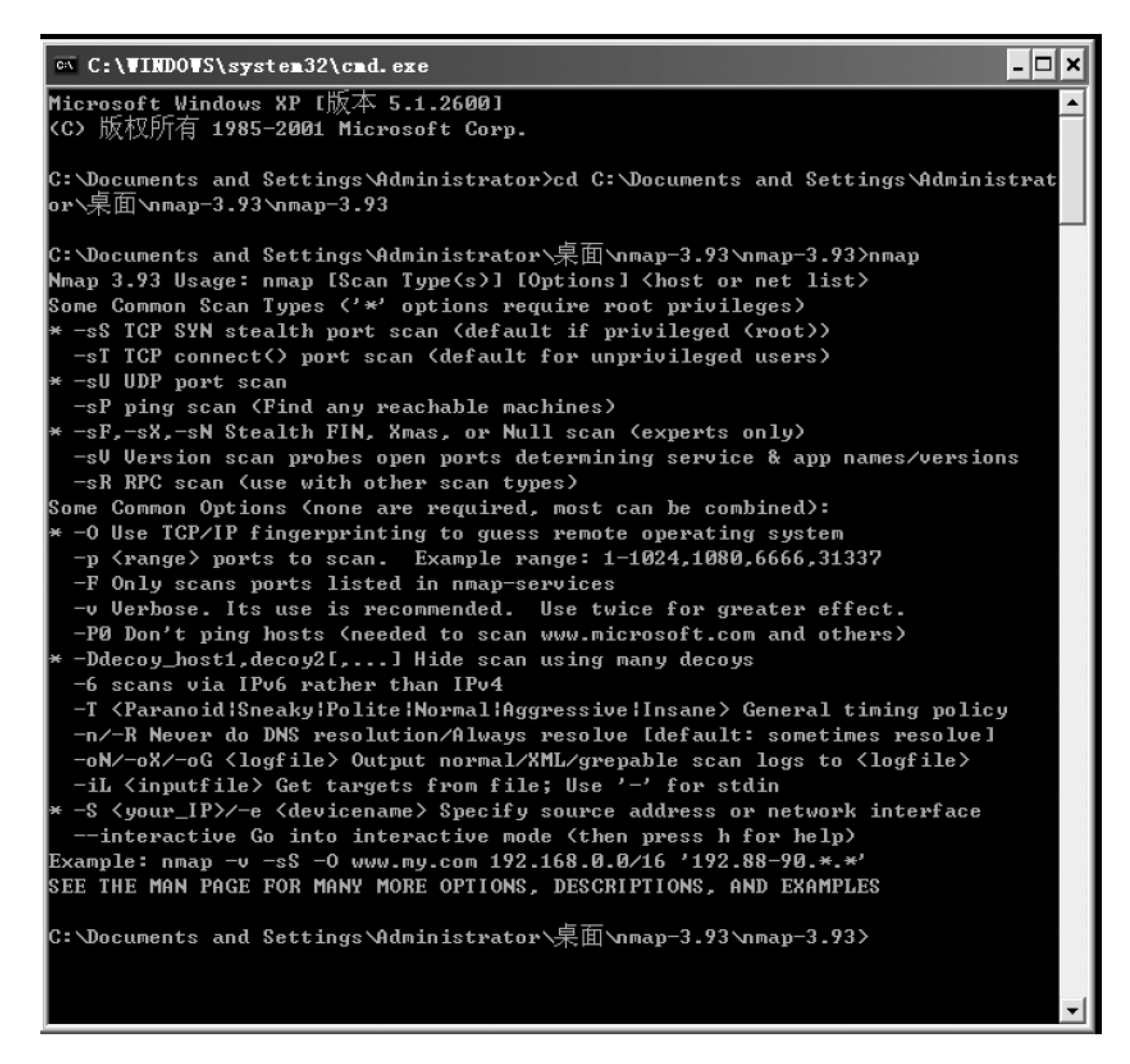


图 10-2　进入 nmap 文件夹

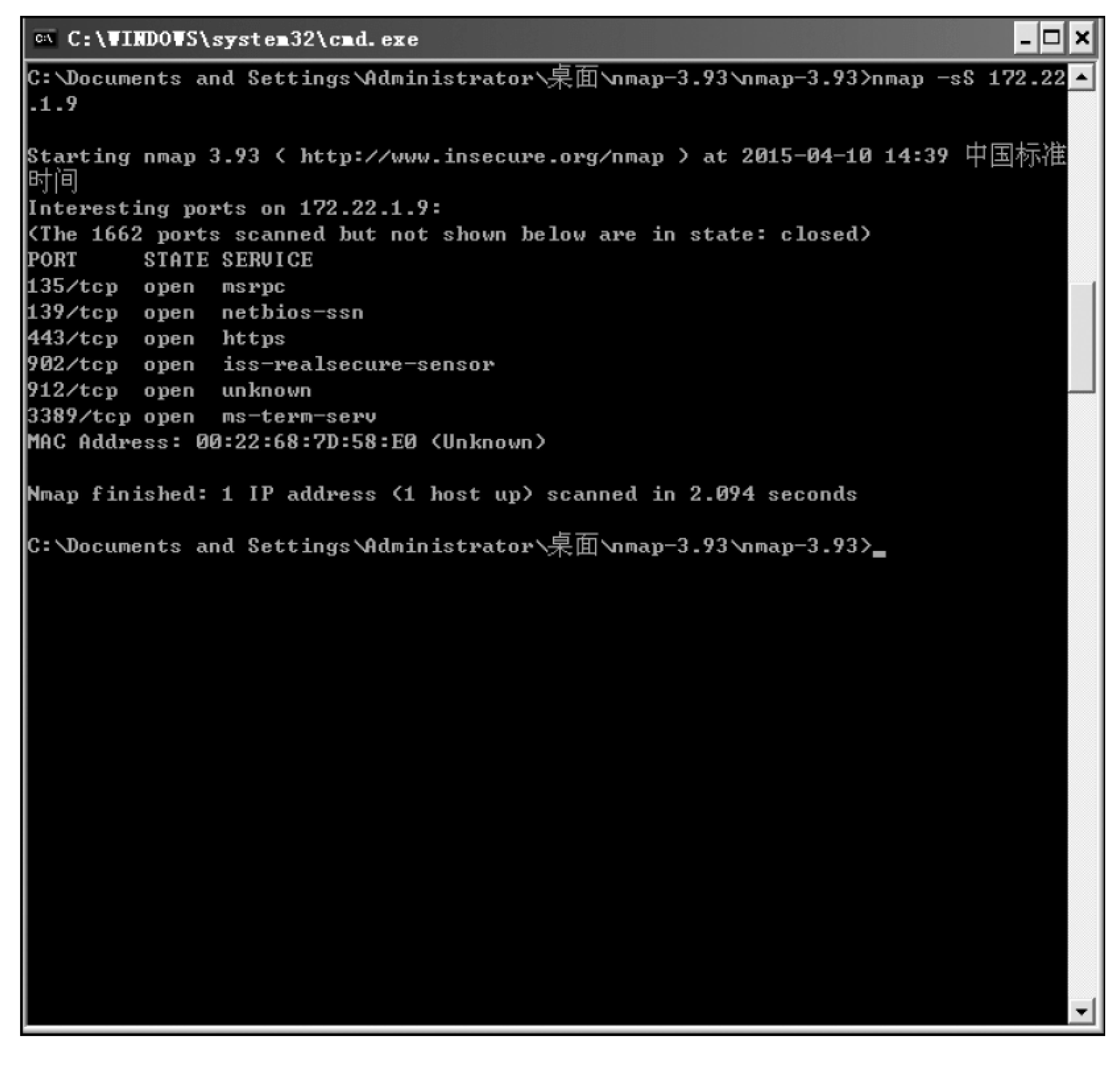
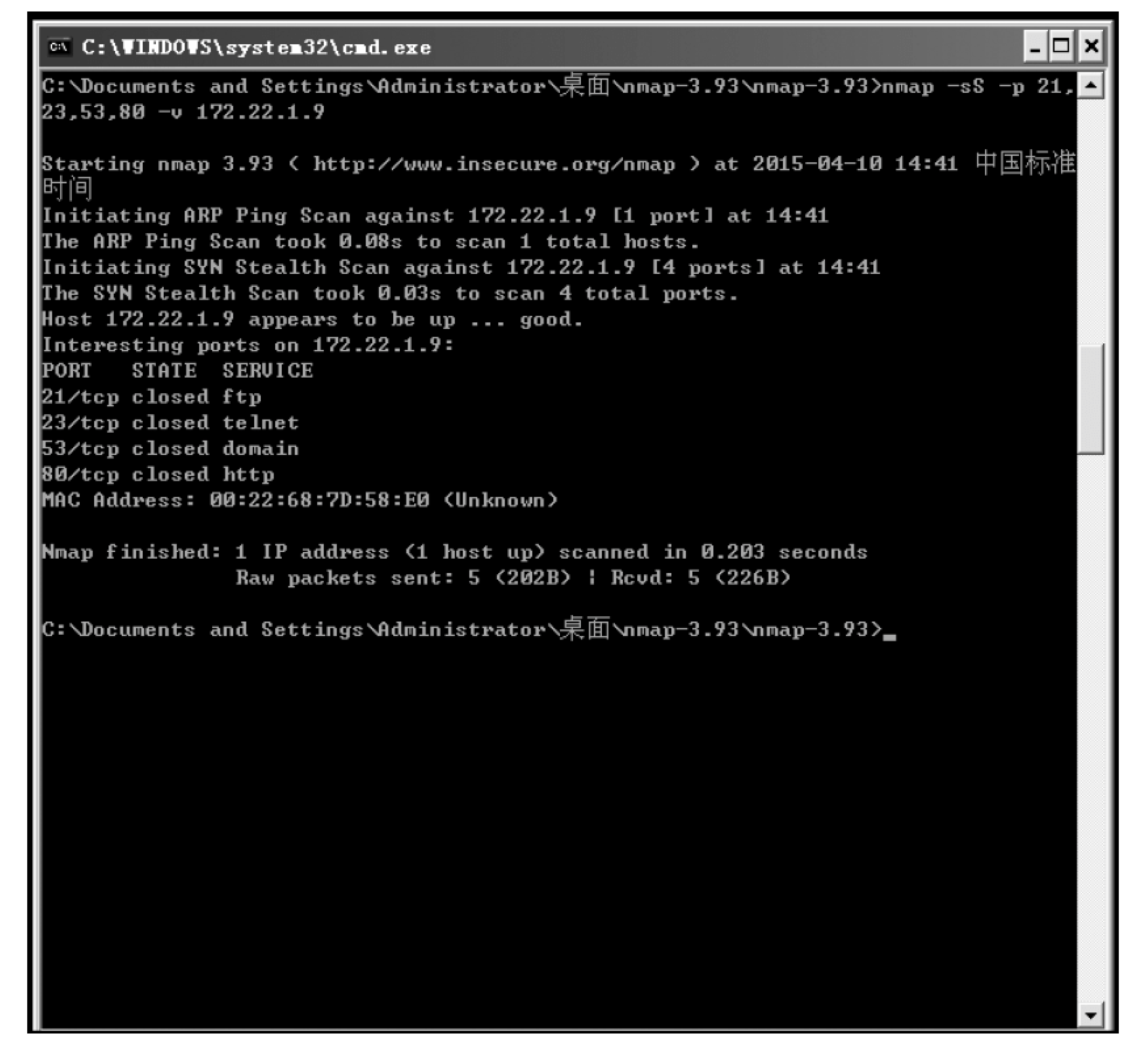


图 10-3　扫描 172.22.1.9 所在网段局域网


```
C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\Administrator\桌面\nmap-3.93\nmap-3.93>nmap -sS -p 21,
23,53,80 -v 172.22.1.9

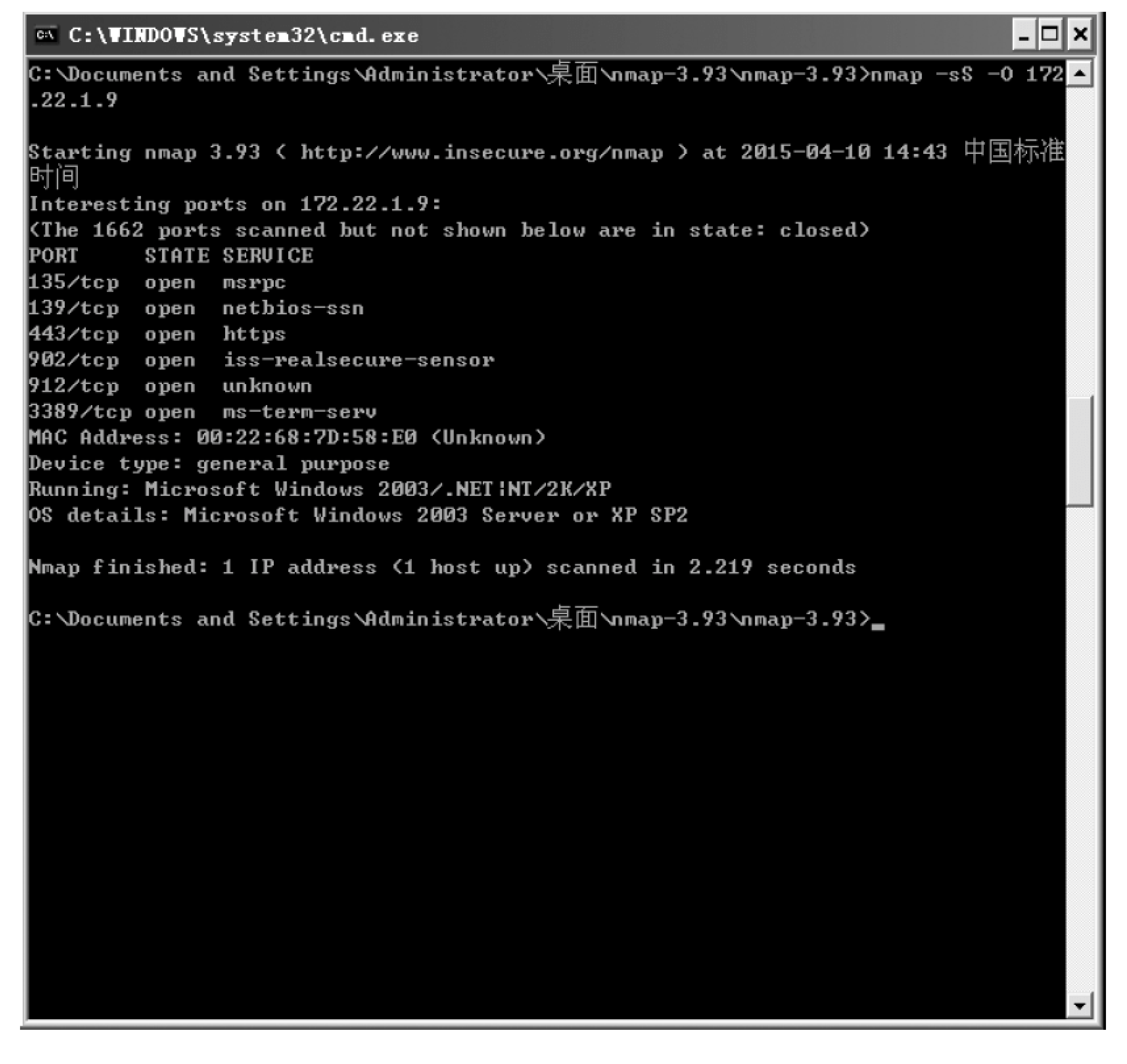
Starting nmap 3.93 ( http://www.insecure.org/nmap ) at 2015-04-10 14:41 中国标准
时间
Initiating ARP Ping Scan against 172.22.1.9 [1 port] at 14:41
The ARP Ping Scan took 0.08s to scan 1 total hosts.
Initiating SYN Stealth Scan against 172.22.1.9 [4 ports] at 14:41
The SYN Stealth Scan took 0.03s to scan 4 total ports.
Host 172.22.1.9 appears to be up ... good.
Interesting ports on 172.22.1.9:
PORT   STATE  SERVICE
21/tcp closed ftp
23/tcp closed telnet
53/tcp closed domain
80/tcp closed http
MAC Address: 00:22:68:7D:58:E0 (Unknown)

Nmap finished: 1 IP address (1 host up) scanned in 0.203 seconds
               Raw packets sent: 5 (202B) | Rcvd: 5 (226B)

C:\Documents and Settings\Administrator\桌面\nmap-3.93\nmap-3.93>_
```

图 10-4　扫描指定端口

（4）操作系统信息扫描：输入命令行 nmap -sS -O 172.22.1.9，如图 10-5 所示。


```
C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\Administrator\桌面\nmap-3.93\nmap-3.93>nmap -sS -O 172
.22.1.9

Starting nmap 3.93 ( http://www.insecure.org/nmap ) at 2015-04-10 14:43 中国标准
时间
Interesting ports on 172.22.1.9:
(The 1662 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
443/tcp  open  https
902/tcp  open  iss-realsecure-sensor
912/tcp  open  unknown
3389/tcp open  ms-term-serv
MAC Address: 00:22:68:7D:58:E0 (Unknown)
Device type: general purpose
Running: Microsoft Windows 2003/.NET|NT/2K/XP
OS details: Microsoft Windows 2003 Server or XP SP2

Nmap finished: 1 IP address (1 host up) scanned in 2.219 seconds

C:\Documents and Settings\Administrator\桌面\nmap-3.93\nmap-3.93>_
```

图 10-5　操作系统信息扫描

（5）Nmap 还提供了欺骗扫描，就是将本地 IP 地址伪装成其他的 IP，如使用 1.1.1.1、2.2.2.2 和 3.3.3.3 作为源地址对目标主机 172.22.1.9 进行 TCP connect()扫描的命令如下：nmap － v － D 1.1.1.1,2.2.2.2,3.3.3.3 － sT 172.22.1.9，结果如图 10-6 所示。

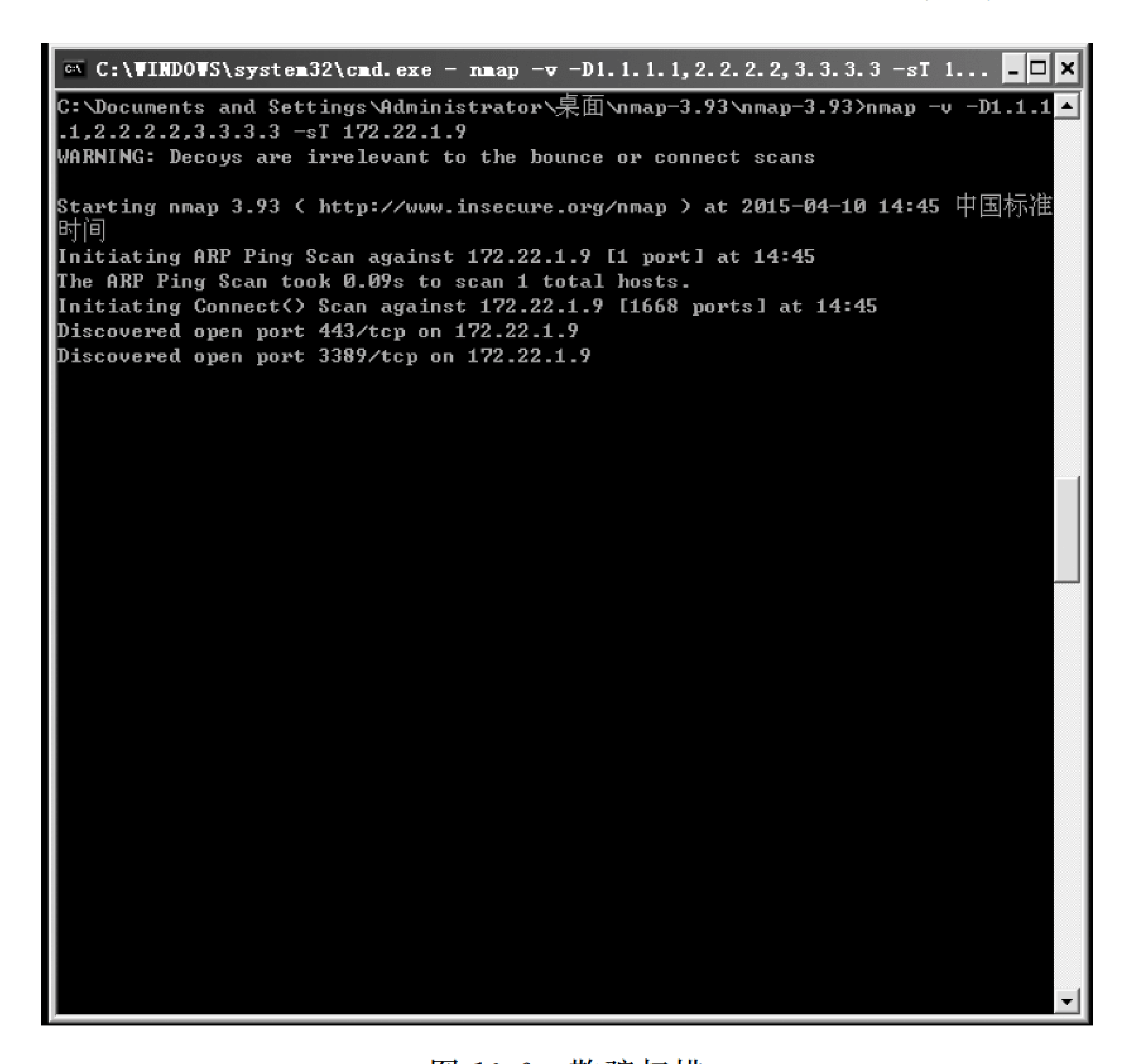


图 10-6　欺骗扫描

除此之外，还可以根据 Nmap 语法规则来测试其他命令的使用，这里不再一一介绍。

10.2　安全漏洞信息探测

10.2.1　安全漏洞探测原理

入侵者在进行攻击之前，首先要收集目标主机的各种信息，找出目标主机的安全漏洞，为后面的继续攻击提供服务。本实验主要通过网络来检测远程目标网络和主机存在的漏洞。其中端口扫描是检测漏洞的主要技术手段。端口是 TCP 协议中定义的，TCP 协议通过套接字(socket)建立起两台计算机之间的网络连接。它采用“IP 地址：端口号”形式定义，通过套接字中不同的端口号来区别同一台计算机上开启的不同 TCP 和 UDP 连接进程。端口是入侵的通道。下面分别介绍几种常用的端口扫描。

10.2.2　端口扫描技术分类

1. TCP connect Scan

TCP connect()扫描：这是最基本的 TCP 扫描方式。扫描主机通过 TCP/IP 协议的

三次握手与目标主机建立一次完整的连接。主机的连接由系统的Connect开始，如果端口开放则连接将建立成功。这种扫描方法的实现简单，对操作者的权限也没有限制。不过这种扫描很容易被检测到，在目标主机的日志中会记录大批的连接请求以及错误信息。

2. TCP SYN Scan

SYN扫描也叫做半连接扫描，主要利用了三次握手协议的弱点，实现方法是向远端主机某端口发送一个只有SYN标志位的TCP数据报，如果主机反馈一个SYN || ACK数据包，那么，这个主机正在监听该端口，如果反馈的是RST数据包，说明，主机没有监听该端口。

3. TCP FIN Scan

对某端口发送一个TCP FIN数据报给远端主机。如果主机没有任何反馈，那么这个主机是存在的，而且正在监听这个端口；主机反馈一个TCP RST回来，说明该主机是存在的，但是没有监听这个端口。由于FIN是中断连接的数据报文，很多设备将SYN数据报过滤，但是FIN数据报能够通过。

4. IP Scan

这种方法并不是直接发送TCP协议探测数据包，而是将数据包分成两个较小的IP协议段。这样就将一个TCP协议头分成好几个数据包，从而过滤器就很难探测到。但必须小心，一些程序在处理这些小数据包时会有些麻烦。

5. TCP Xmas Tree Scan

这种方法向目标端口发送一个含有FIN(结束)，URG(紧急)和PUSH(弹出)标志的分组。根据RFC793，对于所有关闭的端口，目标系统应该返回RST标志。

6. TCP Null Scan

空扫描指主机向目标主机发送一个没有任何标志位的TCP包，根据RFC793，如果目标主机的相应端口是关闭的话，应该发送回一个RST数据包。

10.2.3 X-scan 工具介绍

X-scan被称为速度之王，X-scan采用多线程方式对指定IP地址段(或单机)进行安全漏洞检测，支持插件功能，提供了图形界面和命令行两种操作方式，扫描内容包括：远程操作系统类型及版本，标准端口状态及端口BANNER信息，CGI漏洞，IIS漏洞，RPC漏洞，SQL-SERVER、FTP-SERVER、SMTP-SERVER、POP3-SERVER、NT-SERVER弱口令用户等。X-scan命令语法如下：

```
xscan -host <起始 IP>[-<终止 IP>] <检测项目>[其他选项], xscan -file <主机列表文件名><检测项目>[其他选项]
```

X-scan重要参数解释。

首先介绍检测项目中参数的含义。

-tracert：跟踪路由信息；

-port：检测常用服务端口的状态；

-snmp：检测 Snmp 信息；

-rpc：检测 RPC 漏洞；

-sql：检测 SQL-Server 弱口令)；

-ftp：检测 FTP 弱口令；

-ntpass：检测 NT-Server 弱口令；

-netbios：检测 Netbios 信息；

-smtp：检测 SMTP-Server 漏洞；

-pop3：检测 POP3-Server 弱口令；

-cgi：检测 CGI 漏洞；

-iis：检测 IIS 漏洞；

-bind：检测 BIND 漏洞；

-finger：检测 Finger 漏洞；

-sygate：检测 sygate 漏洞；

-all：检测以上所有项目。

下面介绍部分其他选项的含义：

-v：显示详细扫描进度；

-p：跳过 Ping 不通的主机；

-o：跳过没有检测到开放端口的主机；

-t <并发线程数量[,并发主机数量]>：指定最大并发线程数量和并发主机数量，默认数量为 100,10。

10.2.4　实验 2　X-scan 的使用

【实验目的】

(1) 掌握安全漏洞扫描的原理；

(2) 学习使用工具 X-scan。

【实验内容】

利用 X-scan 扫描器对靶机漏洞进行扫描、分析。

【实验环境】

X-scan 扫描工具、主机一台、靶机一台。

【实验参考步骤】

(1) 配置扫描参数，先单击扫描参数，在“指定 IP 范围”文本框内输入你要扫描主机的 IP 地址，此处使用 IP 地址 192.168.1.135，如图 10-7 所示。

(2) 选择需要扫描的项目，单击扫描模块可以选择扫描的项目。为了大幅度提高扫描的效率，这里选择跳过 Ping 不通的主机，跳过没有开放端口的主机。其他如“端口相关设置”，如图 10-8 所示。

还可以进行如扫描某一特定端口等特殊操作(X-scan 默认也只是扫描一些常用端口)。需要注意的是，在其他设置里，需要选择“无条件扫描”，才可以突破防火墙屏蔽



图 10-7 扫描参数设置

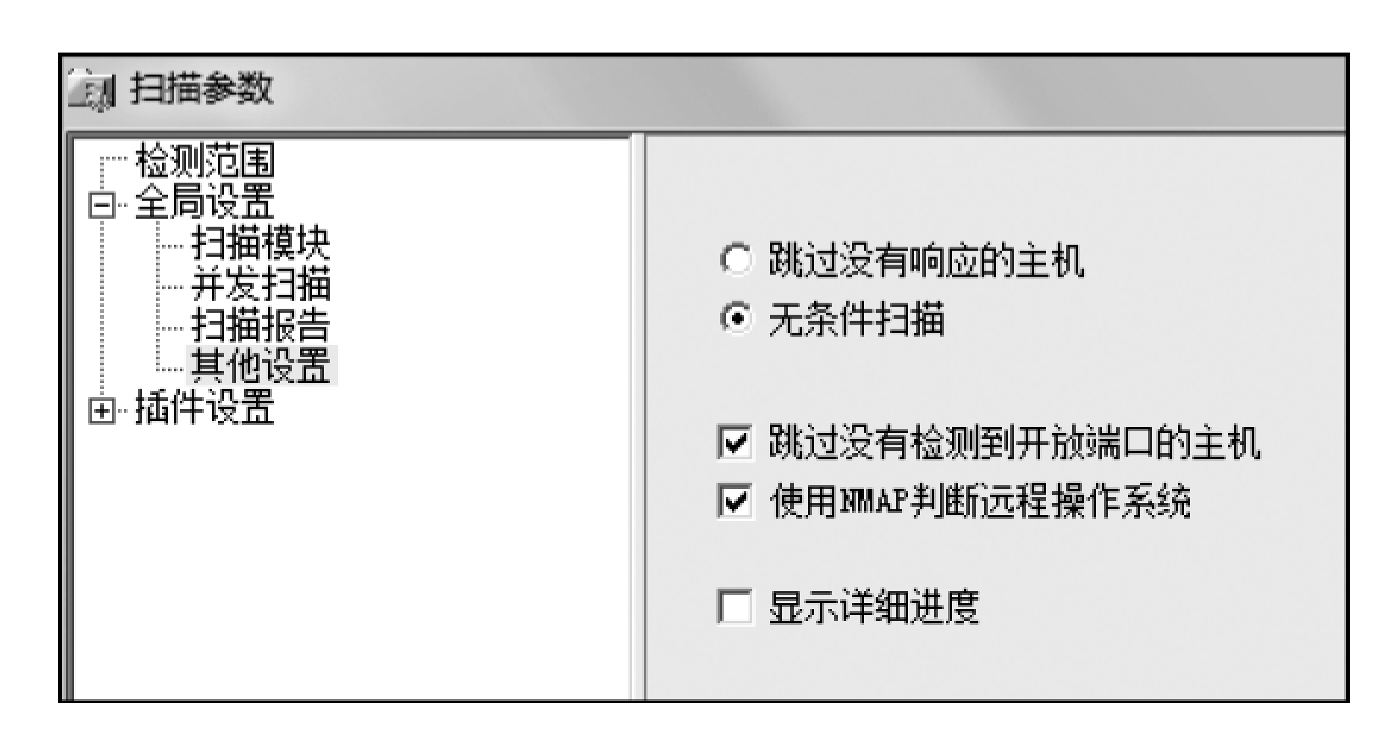


图 10-8 选择扫描项目

ping,进行端口扫描。

(3) 开始扫描,此过程会比较长。扫描结束后会自动生成检测报告,单击“查看”按钮,选择检测报表为 HTML 格式,如图 10-9 所示。

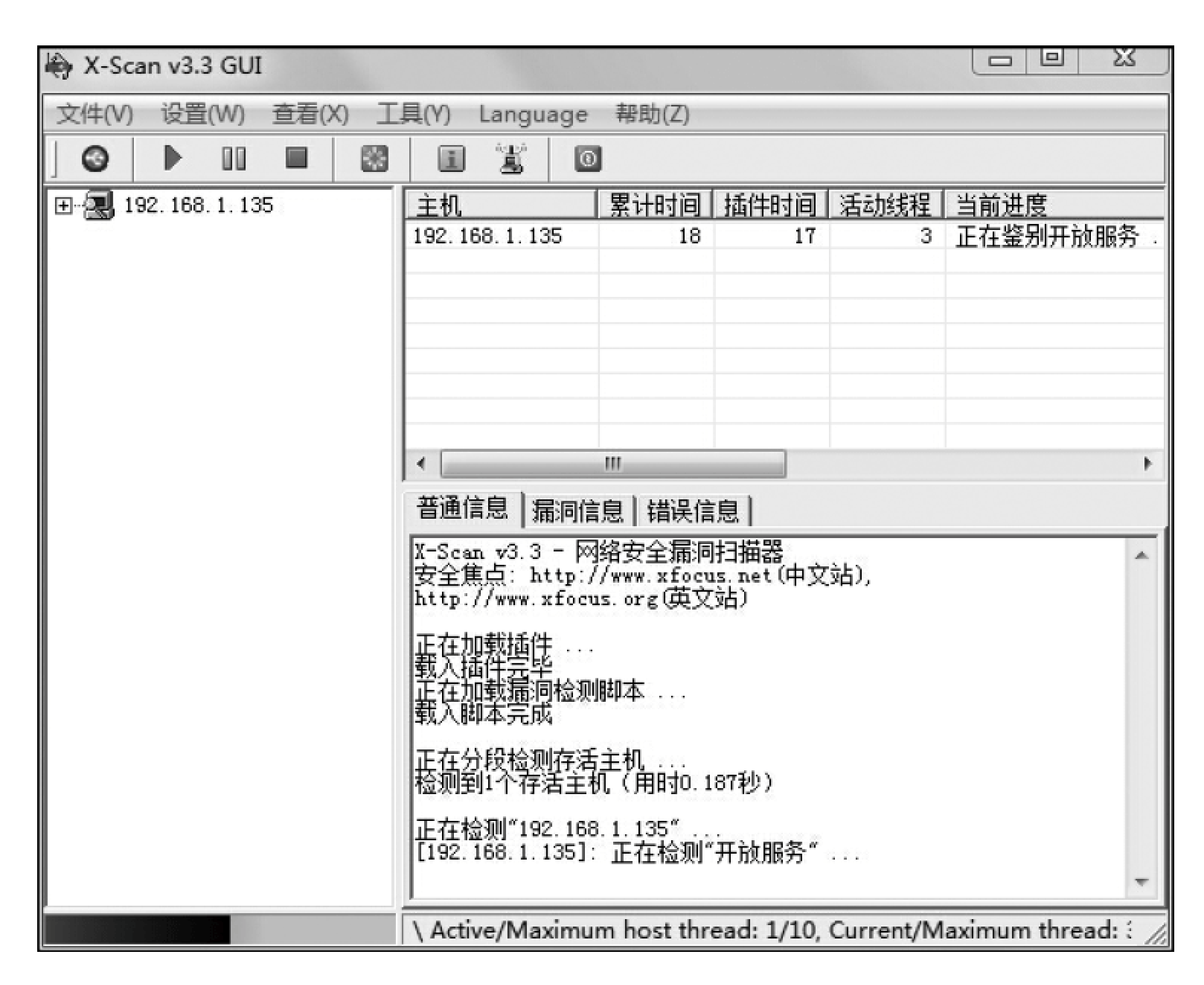


图 10-9 开始扫描

(4) 生成报表，查看靶机中的漏洞数量以及具体的漏洞。从扫描结果可以看出，靶机服务器存在大量的安全漏洞，如图 10-10 所示。

本报表列出了被检测主机的详细漏洞信息，请根据提示信息或链接内容进行相应修补. 欢迎参加X-Scan脚本翻译项目

	检测结果
存活主机	1
漏洞数量	5
警告数量	10
提示数量	17

主机列表

主机	检测结果
172.22.3.43	发现安全漏洞

[返回顶部]

主机分析: 172.22.3.43

主机地址	端口/服务	服务漏洞
172.22.3.43	Windows Terminal Services (3389/tcp)	发现安全警告
172.22.3.43	epmap (135/tcp)	发现安全警告
172.22.3.43	network blackjack (1025/tcp)	发现安全提示
172.22.3.43	microsoft-ds (445/tcp)	发现安全提示
172.22.3.43	netbios-ssn (139/tcp)	发现安全漏洞
172.22.3.43	cifs (445/tcp)	发现安全漏洞

类型	端口/服务	安全漏洞及解决方案
警告	Windows Terminal Services (3389/tcp)	The Terminal Services are enabled on the remote host. Terminal Services allow a Windows user to remotely obtain a graphical login (and therefore act as a local user on the remote host). If an attacker gains a valid login and password, he may be able to use this service to gain further access on the remote host. An attacker may also use this service to mount a dictionnary attack against the remote host to try to log in remotely.

图 10-10　报表生成

【实验报告】

(1) 说明端口扫描原理；

(2) 说明 X-scan 使用步骤。

10.3　小结

本章介绍了信息收集的原理，并通过介绍两种信息收集扫描器，使得读者不仅对信息收集的原理有了更加深刻的理解，也使得读者通过实践明白了信息收集的过程，进一步提高读者对信息安全的重视，并能够在平时生活中利用所学去防御一些简单的攻击。

参 考 文 献

[1] 王路群.计算机网络基础及应用[M].北京：电子工业出版社，2012.
[2] P. Engebretson.渗透测试实践指南：必知必会的工具与方法(原书第2版)[M].姚军，姚明，译.北京：机械工业出版社，2014.
[3] M. Shema. Web应用漏洞侦测与防御[M].齐宁，庞建民等，译.北京：机械工业出版社，2014.
[4] 吴世忠，郭涛，董国伟等.软件漏洞分析技术[M].北京：科学出版社，2014.
[5] 韩启龙.网络安全实验教程(第2版)[M].北京：清华大学出版社，2014.
[6] 周继菲.计算机网络安全技术实验教程[M].北京：北京邮电大学出版社，2009.
[7] 石志国，薛为民等.计算机网络安全教程实验指导[M].北京：北京交通大学出版社，2011.